next up previous

Next: Introduction

User's Manual for `elegant`

Program Version 14.1
Advanced Photon Source
Michael Borland

April 10, 2000

- Introduction
  - Program Philosophy
  - Capabilities of `elegant`
- Namelist Command Dictionary
  - alter_elements
  - amplification_factors
  - analyze_map
  - bunched_beam
  - chromaticity
  - closed_orbit
  - correct
  - correction_matrix_output
  - correct_tunes
  - error_element
  - error_control
  - find_aperture
  - floor_coordinates
  - link_control
  - link_elements
  - load_parameters
  - matrix_output
  - optimize

- optimization_constraint
- optimization_covariable
- optimization_setup
- optimization_term
- optimization_variable
- print_dictionary
- rpn_expression
- run_control
- run_setup
- sasefel
- save_lattice
- sdds_beam
- subprocess
- steering_element
- twiss_output
- track
- vary_element

- Accelerator and Element Description
- Element Dictionary
  - ALPH
  - BMAPXY
  - BUMPER
  - CENTER
  - CEPL
  - CHARGE
  - CSBEND
  - CSRCSBEND
  - CSRDRIFT
  - DRIF
  - ECOL
  - ELSE
  - ENERGY
  - FMULT

- HISTOGRAM
- HKICK
- HMON
- IBSCATTER
- KICKER
- KPOLY
- KQUAD
- KSBEND
- KSEXT
- MAGNIFY
- MALIGN
- MARK
- MATR
- MATTER
- MAXAMP
- MODRF
- MONI
- MULT
- NIBEND
- NISEPT
- OCTU
- PEPPOT
- PFILTER
- QUAD
- QUFRINGE
- RAMPP
- RAMPRF
- RBEN
- RCOL
- RECIRC
- RFCA
- RFCW
- RFDF

- RFMODE
- RFTM
- RMDF
- ROTATE
- SAMPLE
- SBEN
- SCATTER
- SCRAPER
- SEXT
- SOLE
- SREFFECTS
- STRAY
- TMCF
- TRCOUNT
- TRFMODE
- TRWAKE
- TUBEND
- TWLA
- TWMTA
- TWPL
- VKICK
- VMON
- WAKE
- WATCH
- ZLONGIT
- ZTRANSVERSE

- Examples
- The `rpn` Calculator
- Bibliography
- About this document ...

Michael Borland 2000-04-10

Introduction

`elegant` stands for ``ELEctron Generation ANd Tracking,'' a somewhat out-of-date description of a fully 6D accelerator program that now does much more than generate particle distributions and track them. `elegant`, written entirely in the C programming language[1], uses a variant of the MAD[2] input format to describe accelerators, which may be either transport lines, circular machines, or a combination thereof. Program execution is driven by commands in a namelist format.

This document describes the features available in `elegant`, listing the commands and their arguments. The differences between `elegant` and MAD formats for describing accelerators are listed. A series of examples of `elegant` input and output are given. Finally, appendices are included describing the post-processing programs.

Subsections

- Program Philosophy
- Capabilities of `elegant`

Michael Borland 2000-04-10

Program Philosophy

For all its complexity, `elegant` is not a stand-alone program. For example, most of the output is not human-readable, and `elegant` itself has no graphics capabilities. These tasks are handled by a suite of post-processing programs that serve both `elegant` and other physics programs. These programs, collectively known as the SDDS Toolkit[8,9], provide sophisticated data analysis and display capabilities. They also serve to prepare input for `elegant`, supporting multi-stage simulation.

Setting up for an `elegant` run thus involves more than creating input files for `elegant` per se. A complicated run will typically involve creation of a post-processing command file that processes `elegant` output and puts it in the most useful form, typically a series of graphs. Users thus have the full power of the SDDS Toolkit, the resident command interpreter (e.g., the UNIX shell), and their favorite scripting language (e.g., Tcl/Tk) at their disposal. The idea is that instead of continually rewriting the physics code to, for example, make another type of graph or squeeze another item into a crowded table, one should allow the user to tailor the output to his specific needs using a set of generic post-processing programs. This approach has been quite successful, and is believed particularly suited to the constantly changing needs of research.

Unlike many other programs, `elegant` allows one to make a single run simulating an arbitrary number of randomizations or variations of an accelerator. By using the SDDS toolkit to postprocess the data, the user's postprocessing time and effort do not depend on how many random seeds or situations are chosen. Hence, instead of doing a few simulations with a few seed numbers or values, the user can simulate hundreds or even thousands of instances of one accelerator to get an accurate representation of the statistics or dependence on parameters, with no more work invested than in doing a few simulations.

In addition, complex simulations such as top-up tracking[11] can be performed involving thousands of runs, with input created by scripts depending on the SDDS toolkit. These simulations make use of concurrent computing on about 20 workstation using the Distributed Queueing System[10]. Clearly, use of automated postprocessing tools greatly increases the scale and sophistication of simulations possible. This stands in stark contrast to the current trend toward graphical user interfaces, which virtually force an inefficient one-job, one-computer, manual postprocessing way of working.

Bibliography

1

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J., second edition, 1988.

2

H. Grote, F. C. Iselin, ``The MAD Program-Version 8.1," CERN/SL/90-13(AP), June 1991.

3

K. L. Brown, R. V. Servranckx, ``First- and Second-Order Charged Particle Optics," SLAC-PUB-3381, July 1984.

4

M. Borland, ``A High-Brightness Thermionic Microwave Electron Gun," SLAC-Report-402, February 1991, Stanford University Ph.D. Thesis.

5

H. A. Enge, ``Achromatic Mirror for Ion Beams," Rev. Sci. Inst., 34(4), 1963.

6

M. Borland, private communication.

7

W. H. Press, *et al*, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.

8

M. Borland, ``A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," Proc. 1995 PAC, May 1-5, 1995, Dallas, Texas, pp. 2184-2186 (1996).

9

M. Borland, ``A Universal Postprocessing Toolkit for Accelerator Simulation and Data Analysis," Proc. 1998 ICAP Conference, Sept. 14-18, 1998, Monterey, California, to be published.

10

T. P. Green, ``Research Toward a Heterogeneous Networked Computer Cluster: The Distributed Queuing System Version 3.0," SCRI Technical Publication, 1994.

11

M. Borland and L. Emery, ``Tracking Studies of Top-Up Safety for the Advanced Photon Source,", Proceedings of the 1999 Particle Accelerator Conference, New York, 1999, pg 2319-2321.

12

M. Xie, ``Free ElectronLaser Driven by SLAC LINAC".

Michael Borland 2000-04-10

Capabilities of `elegant`

`elegant` started as a tracking code, and it is still well-suited to this task. `elegant` tracks in the 6-dimensional phase space $(x, x', y, y', s, \delta)$, where x (y) is the horizontal (vertical) transverse coordinate, primed quantities are slopes, s is the *total* distance traveled, and δ is the fractional momentum deviation[3]. Note that these quantities are commonly referred to as (x, xp, y, yp, s, dp) in the namelists, accelerator element parameters, and output files. (``dp" is admittedly confusing--it is supposed to remind the user of $\Delta P/P_o$. Sometimes this quantity is referred to as ``delta.")

Tracking may be performed using matrices (of selectable order), canonical kick elements, numerically integrated elements, or any combination thereof. For most elements, second-order matrices are available; matrix concatenation can be done to any order up to third. Canonical kick elements are available for bending magnets, quadrupoles, sextupoles, and higher-order multipoles; all of these elements also support optional classical synchrotron radiation losses. Among the numerically integrated elements available are extended-fringe-field bending magnets and traveling-wave accelerators. A number of hybrid elements exist that have first-order transport with exact time dependence, e.g., RF cavities. Several elements support simulation of collective effects, such as wakefields and intra-beam scattering. Some of the more unusual elements available are third-order alpha-magnets[4,5], time-dependent kicker magnets, voltage-ramped RF cavities, beam scrapers, and beam-analysis ``screens."

A wide variety of output is available from tracking, including centroid and sigma-matrix output along the accelerator. In addition to tracking internally generated particle distributions, `elegant` can track distributions stored in external files, which can either be generated by other programs or by previous `elegant` runs. Because `elegant` uses SDDS format for reading in and writing out particle coordinates, it is relatively easy to interface `elegant` to other programs using files that can also be used with SDDS to do post-processing for the programs.

`elegant` allows the addition of random errors to virtually any parameter of any accelerator element. One can correct the orbit (or trajectory), tunes, and chromaticity after adding errors, then compute Twiss parameters, track, or perform a number of other operations.

In addition to randomly perturbing accelerator elements, `elegant` allows one to systematically vary any number of elements in a multi-dimensional grid. As before, one can track or do other computations for each point on the grid. This is a very useful feature for the simulation of experiments, e.g., emittance measurements involving beam-size measurements during variation of one or more quadrupoles[6].

Like many accelerator codes, `elegant` does accelerator optimization. While `elegant` is not designed to replace matching codes like MAD, it will fit the first-order matrix and beta functions. Of more interest is the ability to optimize results of tracking using a user-supplied function of the final beam and transport

parameters. This permits solution of a wide variety of problems, from matching a kicker bump in the presence of nonlinearities to optimizing dynamic aperture by adjusting sextupoles.

`elegant` provides several methods for determining accelerator aperture, whether dynamic or physical. One may do straightforward tracking of an ensemble of particles that occupies at uniform grid in (x, y) space. A more efficient variant of this procedure involves tracking a series of constant-x lines of particles with fixed y values, with elimination of any given y value whenever a stable particle is found. Finally, one may use a single-particle search method that can locate the aperture for a series of y values, to a predefined resolution in x.

In addition to using analytical expressions for the transport matrices, `elegant` supports computation of the first-order matrix and linear optics properties of a circular machine based on tracking. A common application of this is to compute the tune and beta-function variation with momentum offset by single-turn tracking of a series of particles. This is much more efficient than, for example, tracking and performing FFTs (though `elegant` will do this also). This both tests analytical expressions for the chromaticity and allows computations using accelerator elements for which such expressions do not exist (e.g., a numerically integrated bending magnet with extended fringe fields).

A common application of random error simulations is to set tolerances on magnet strength and alignment relative to the correctability of the closed orbit. A more efficient way to do these calculations is to use correct-orbit amplification factors[6]. `elegant` the computes amplification factors and functions for corrected and uncorrected orbits and trajectories pertaining to any element that produces an orbit or trajectory distortion. It simultaneously computes the amplification functions for the steering magnets, in order to determine how strong the steering magnets will need to be.

---

next up previous

Next: Namelist Command Dictionary **Up:** Introduction **Previous:** Program Philosophy Michael Borland 2000-04-10

Namelist Command Dictionary

The main input file for an `elegant` run consists of a series of namelists, which function as commands. Most of the namelists direct `elegant` to set up to run in a certain way. A few are ``action'' commands that begin the actual simulation. FORTRAN programmers should note that, unlike FORTRAN namelists, these namelists need not come in a predefined order; `elegant` is able to detect which namelist is next in the file and react appropriately.

Each namelist has a number of variables associated with it, which are used to control details of the run. These variables come in three data types: (1) `long`, for the C long integer type. (2) `double`, for the C double-precision floating point type. (3) `STRING`, for a character string enclosed in double quotation marks. All variables have default values, which are listed on the following pages. `STRING` variables often have a default value listed as `NULL`, which means no data; this is quite different from the value ``'', which is a zero-length character string. `long` variables are often used as logical flags, with a zero value indicating false and a non-zero value indicating true.

On the following pages the reader will find individual descriptions of each of the namelist commands and their variables. Each description contains a sequence of the form

```
&<namelist-name>
    <variable-type> <variable-name> = <default-value>;
    .
    .
    .
&end
```

This summarizes the parameters of the namelist. Note, however, that the namelists are invoked in the form

```
&<namelist-name>
    [<variable-name> = <value> ,]
    [<array-name>[<index>] = <value> [,<value> ...] ,]
        .
        .
        .
&end
```

The square-brackets enclose an optional component. Not all namelists require variables to be given-the defaults may be sufficient. However, if a variable name is given, it must have a value. Values for `STRING` variables must be enclosed in double quotation marks. Values for `double` variables may be in floating-point, exponential, or integer format (exponential format uses the `e' character to introduce the

exponent).

Array variables take a list of values, with the first value being placed in the slot indicated by the subscript. As in C, the first slot of the array has subscript 0, *not* 1. The namelist processor does not check to ensure that you do not put elements into nonexistent slots beyond the end of the array; doing so may cause the processor to hang up or crash.

Wildcards are allowed in a number of places in `elegant` and the SDDS Toolkit. The wildcard format is very similar to that used in UNIX:

- `*` -- stands for any number of characters, including none.
- `?` -- stands for any single character.
- `[<list-of-characters>]` -- stands for any single character from the list. The list may include ranges, such as `a-z`, which includes all characters between and including `a' and `z' in the ASCII character table.

The special characters `*`, `?`, `[`, and `]` are entered literally by preceeding the character by a backslash (e.g., `\*`).

In many places where a filename is required in an `elegant` namelist, the user may supply a so-called ``incomplete" filename. An incomplete filename has the sequence ``%s" imbedded in it, for which is substituted the ``rootname." The rootname is by default the filename (less the extension) of the lattice file. The most common use of this feature is to cause `elegant` to create names for all output files that share a common filename but differ in their extensions. Post-processing can be greatly simplified by adopting this naming convention, particularly if one consistently uses the same extension for the same type of output. Recommended filename extensions are given in the lists below.

When `elegant` reads a namelist command, one of its first actions is to print the namelist back to the standard output. This printout includes all the variables in the namelist and their values. Occasionally, the user may see a variable listed in the printout that is not in this manual. These are often obsolete and are retained only for backward compatibility, or else associated with a feature that is not fully supported. Use of such ``undocumented features" is discouraged.

Subsections

- alter_elements
- amplification_factors
- analyze_map
- bunched_beam
- chromaticity
- closed_orbit
- correct
- correction_matrix_output
- correct_tunes

- error_element
- error_control
- find_aperture
- floor_coordinates
- link_control
- link_elements
- load_parameters
- matrix_output
- optimize
- optimization_constraint
- optimization_covariable
- optimization_setup
- optimization_term
- optimization_variable
- print_dictionary
- rpn_expression
- run_control
- run_setup
- sasefel
- save_lattice
- sdds_beam
- subprocess
- steering_element
- twiss_output
- track
- vary_element

---

next up previous

Next: alter_elements **Up:** User's Manual for elegant **Previous:** Capabilities of elegant Michael Borland 2000-04-10

alter_elements

- type: action command.
- function: modify the value of a parameter for one or more elements

```
&alter_elements
        STRING name = NULL;
        STRING item = NULL;
        STRING exclude = NULL;
        double value = 0;
        long differential = 0;
        long multiplicative = 0;
        long verbose = 0;
        long allow_missing_parameters = 0;
&end
```

- `name` -- A possibly-wildcard-containing string giving the names of the elements to alter.
- `item` -- The name of the parameter to alter.
- `item` -- A possibly-wildcard-containing string giving the names of elements to excluded from alteration.
- `value` -- The new value for the parameter.
- `differential` -- If nonzero, the new value is the predefined value of the parameter plus the quantity given with `value`.
- `multiplicative` -- If nonozero, the new given value is the predefined value of the parameter times the quantity given with `value`.
- `verbose` -- If nonzero, information is printed to the standard output describing what elements are changed.
- `allow_missing_parameters` -- If nonzero, then it is not an error if an element does not have the parameter named with `item`. Normally, such an occurence is an error and terminates the program.

Michael Borland 2000-04-10

amplification_factors

- type: action command.
- function: compute corrected and uncorrected orbit amplification factors and functions.

```
&amplification_factors
    STRING output = NULL;
    STRING uncorrected_orbit_function = NULL;
    STRING corrected_orbit_function = NULL;
    STRING kick_function = NULL;
    STRING name = NULL;
    STRING type = NULL;
    STRING item = NULL;
    STRING plane = NULL;
    double change = 1e-3;
    long number_to_do = -1;
    double maximum_z = 0;
&end
```

- `output` -- The (incomplete) name of a file for text output. Recommended value: ``%s.af''.
- `uncorrected_orbit_function` -- The (incomplete) name of a file for an `mpl`-format output of the uncorrected-orbit amplification function. Recommended value: ``%s.uof''.
- `corrected_orbit_function` -- The (incomplete) name of a file for an `mpl`-format output of the corrected-orbit amplification function. Recommended value: ``%s.cof''.
- `kick_function` -- The (incomplete) name of a file for an `mpl`-format output of the kick amplification function. Recommended value: ``%s.kaf''.
- `name` -- The optionally wildcarded name of the orbit-perturbing elements.
- `type` -- The optional type name of the the orbit-perturbing elements.
- `item` -- The parameter of the elements producing the orbit.
- `plane` -- The plane (``h'' or ``v'') to examine.
- `change` -- The parameter change to use in computing the amplification.
- `number_to_do` -- The number of elements to perturb.
- `maximum_z` -- The maximum z coordinate of the elements to perturb.

Michael Borland 2000-04-10

analyze_map

- type: action command.
- function: find the approximate first-order matrix and related quantities for an accelerator by tracking.

```
&analyze_map
    STRING output = NULL;
    double delta_x = 1e-6;
    double delta_xp = 1e-6;
    double delta_y = 1e-6;
    double delta_yp = 1e-6;
    double delta_s  = 1e-6;
    double delta_dp = 1e-6;
    long center_on_orbit = 0;
    long verbosity = 0;
&end
```

- `output` -- The (incomplete) name of a file for SDDS output.
  - Recommended value: ``%s.ana''.
  - File contents: A series of dumps, each consisting of a single data point containing the centroid offsets for a single turn, the single-turn R matrix, the matched Twiss parameters, tunes, and dispersion functions.
- `delta_X` -- The amount by which to change the quantity X in computing the derivatives that give the matrix elements.
- `center_on_orbit` -- A flag directing the expansion to be made about the closed orbit instead of the design orbit.
- `verbosity` -- The larger this value, the more output is printed during computations.

Michael Borland 2000-04-10

bunched_beam

- type: setup command.
- function: set up for tracking of particle coordinates with various distributions.

```
&bunched_beam
    STRING bunch = NULL;
    long n_particles_per_bunch = 1;
    double time_start = 0;
    STRING matched_to_cell = NULL;
    double emit_x  = 0;
    double beta_x  = 1.0;
    double alpha_x = 0.0;
    double eta_x   = 0.0;
    double etap_x  = 0.0;
    double emit_y  = 0;
    double beta_y  = 1.0;
    double alpha_y = 0.0;
    double eta_y   = 0.0;
    double etap_y  = 0.0;
    long use_twiss_command_values = 0;
    double Po = 0.0;
    double sigma_dp = 0.0;
    double sigma_s = 0.0;
    double dp_s_coupling = 0;
    long one_random_bunch = 1;
    long symmetrize = 0;
    long limit_invariants = 0;
    long limit_in_4d = 0;
    long enforce_rms_values[3] = {0, 0, 0};
    double distribution_cutoff[3] = {2, 2, 2};
    STRING distribution_type[3] = {"gaussian","gaussian","gaussian"};
    double centroid[6] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    long save_initial_coordinates = 1;
&end
```

- `bunch` -- The (incomplete) name of an SDDS file to which the phase-space coordinates of the bunches are to be written. Recommended value: ``%s.bun".
- `n_particles_per_bunch` -- Number of particles in each bunch.
- `time_start` -- The central value of the time coordinate for the bunch.

- `matched_to_cell` -- The name of a beamline from which the Twiss parameters of the bunch are to be computed.
- `emit_X` -- RMS emittance for the X plane.
- `beta_X`, `alpha_X`, `eta_X`, `etap_X` -- Twiss parameters for the X plane.
- `use_twiss_command_values` -- If nonzero, then the values for β, α, η, and η' are taken from the `twiss` command. It is an error if no `twiss` command has been given.
- `Po` -- Central momentum of the bunch.
- `sigma_dp`, `sigma_s` -- Fractional momentum spread, δ, and bunch length.
- `dp_s_coupling` -- Specifies the coupling between s and δ, defined as $\langle s\delta\rangle/(\sigma_s\sigma_\delta)$.
- `one_random_bunch` -- If non-zero, then only one random particle distribution is generated. Otherwise, a new distribution will be generated for every simulation step.
- `enforce_rms_values[3]` -- Flags, one for each plane, indicating whether to force the distribution to have the specified RMS properties.
- `distribution_cutoff[3]` -- Distribution cutoff parameters for each plane.
- `distribution_type[3]` -- Distribution type for each plane. May be ``gaussian'', ``hard-edge'', ``uniform-ellipse'', ``shell'', or ``dynamic-aperture''.
- `limit_invariants` -- If non-zero, the distribution cutoffs are applied to the invariants, rather than to the coordinates.
- `limit_in_4d` -- If non-zero, then the transverse distribution is taken to be a 4-d gaussian or uniform distribution. One of these must be chosen using the `distribution_type` control. It must be the same for x and y.
- `symmetrize` -- If non-zero, the distribution is symmetric under changes of sign in the coordinates. Automatically results in a zero centroid for all coordinates.
- `centroid[6]` -- Centroid offsets for each of the six coordinates.
- `save_initial_coordinates` -- A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are regenerated each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

---

next up previous

Next: chromaticity **Up:** Namelist Command Dictionary **Previous:** analyze_map Michael Borland 2000-04-10

chromaticity

- type: setup command.
- function: set up for chromaticity correction.

```
&chromaticity
    STRING sextupoles = NULL;
    double dnux_dp = 0;
    double dnuy_dp = 0;
    double sextupole_tweek = 1e-3;
    double correction_fraction = 0.9;
    long n_iterations = 5;
    double tolerance = 0;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    double strength_limit = 0;
    long use_perturbed_matrix = 0;
&end
```

- `sextupoles` -- List of names of elements to use to correct the chromaticities.
- `dnux_dp`, `dnuy_dp` -- Desired chromaticity values.
- `sextupole_tweek` -- Amount by which to tweak the sextupoles to compute derivatives of chromaticities with respect to sextupole strength. [The word ``tweak'' is misspelled ``tweek'' in the code.]
- `correction_fraction` -- Fraction of the correction to apply at each iteration. In some cases, correction is unstable at this number should be reduced.
- `n_iterations` -- Number of iterations of the correction to perform.
- `tolerance` -- Stop iterating when chromaticities are within this value of the desired values.
- `strength_log` -- The (incomplete) name of an SDDS file to which the sextupole strengths will be written. Recommended value: ``%s.ssl''.
- `change_defined_values` -- Changes the defined values of the sextupole strengths. This means that when the lattice is saved (using `save_lattice`), the sextupoles will have the corrected values. This would be used for correcting the chromaticity of a design lattice, for example, but not for correcting chromaticity of a perturbed lattice.
- `strength_limit` -- Limit on the absolute value of sextupole strength (K_2).
- `use_perturbed_matrix` -- If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain

correction. In general, it is not necessary and only slows the simulation.

---

next up previous

Next: closed_orbit **Up:** Namelist Command Dictionary **Previous:** bunched_beam Michael Borland 2000-04-10

closed_orbit

- type: setup command.
- function: set up for computation of the closed orbit.

```
&closed_orbit
    STRING output = NULL;
    long output_monitors_only = 0;
    long start_from_centroid = 1;
    double closed_orbit_accuracy = 1e-12;
    long closed_orbit_iterations = 10;
    double iteration_fraction = 1;
    long fixed_length = 0;
    long start_from_recirc = 0;
    long verbosity = 0;
&end
```

- `output` -- The (incomplete) name of an SDDS file to which the closed orbits will be written. Recommended value: ``%s.clo''.
- `output_monitors_only` -- If non-zero, indicates that the closed orbit output should include only the data at the locations of the beam-position monitors.
- `start_from_centroid` -- A flag indicating whether to force the computation to start from the centroids of the beam distribution.
- `closed_orbit_accuracy` -- The desired accuracy of the closed orbit, in terms of the difference between the start and end coordinates, in meters.
- `closed_orbit_iterations` -- The number of iterations to take in finding the closed orbit.
- `iteration_fraction` -- Fraction of computed change that is used each iteration. For lattices that are very nonlinear or close to unstable, a number less than 1 can be helpful. Otherwise, it only slows the simulation.
- `fixed_length` -- A flag indicating whether to find a closed orbit with the same length as the design orbit by changing the momentum offset.
- `start_from_recirc` -- A flag indicating whether to compute the closed orbit from the recirculation (`recirc`) element in the beamline. In general, if you have a recirculation element, you should give this flag.
- `verbosity` -- A larger value results in more printouts during the computations.

Next: correct **Up:** Namelist Command Dictionary **Previous:** chromaticity Michael Borland 2000-04-10

correct

- type: setup command.
- function: set up for correction of the trajectory or closed orbit.

```
&correct
    STRING mode = "trajectory";
    STRING method = "global";
    STRING trajectory_output = NULL;
    STRING corrector_output = NULL;
    STRING statistics = NULL;
    double corrector_tweek[2] = {1e-3, 1e-3};
    double corrector_limit[2] = {0, 0};
    double correction_fraction[2] = {1, 1};
    double correction_accuracy[2] = {1e-6, 1e-6};
    double bpm_noise[2] = {0, 0};
    double bpm_noise_cutoff[2] = {1.0, 1.0};
    STRING bpm_noise_distribution[2] = {"uniform", "uniform"};
    long verbose = 1;
    long fixed_length = 0;
    long fixed_length_matrix = 0;
    long n_xy_cycles = 1;
    long n_iterations = 1;
    long prezero_correctors = 1;
    long track_before_and_after = 0;
    long start_from_centroid = 1;
    long use_actual_beam = 0;
    double closed_orbit_accuracy = 1e-12;
    long closed_orbit_iterations = 10;
    double closed_orbit_iteration_fraction = 1;
&end
```

- `mode` -- Either ``trajectory'' or ``orbit'', indicating correction of a trajectory or a closed orbit.
- `method` -- For trajectories, may be ``one-to-one'' or ``global''. For closed orbit, must be ``global''.
- `trajectory_output` -- The (incomplete) name of an SDDS file to which the trajectories or orbits will be written. Recommended value: ``%s.traj'' or ``%s.orb''.
- `corrector_output` -- The (incomplete) name of an SDDS file to which information about the final corrector strengths will be written. Recommended value: ``%s.cor''.
- `statistics` -- The (incomplete) name of an SDDS file to which statistical information about

the trajectories (or orbits) and corrector strengths will be written. Recommended value: ``%s.scor".

- `corrector_tweek[2]` -- The amount by which to change the correctors in order to compute correction coefficients. [The word ``tweak" is misspelled ``tweek" in the code.]
- `corrector_limit[2]` -- The maximum strength allowed for a corrector.
- `correction_fraction[2]` -- The fraction of the computed correction strength to actually use for any one iteration.
- `correction_accuracy[2]` -- The desired accuracy of the correction in terms of the RMS BPM values.
- `bpm_noise[2]` -- The BPM noise level.
- `bpm_noise_cutoff[2]` -- Cutoff values for the random distributions of BPM noise.
- `bpm_noise_distribution[2]` -- May be either ``gaussian", ``uniform", or ``plus_or_minus".
- `verbose` -- If non-zero, information about the correction is printed during computations.
- `fixed_length` -- Indicates that the closed orbit length should be kept the same as the design orbit length by changing the momentum offset of the beam.
- `fixed_length_matrix` -- Indicates that for fixed-length orbit correction, the fixed-length matrix should be computed and used. This will improve convergence but isn't always needed.
- `n_xy_cycles` -- Number of times to alternate between correcting the x and y planes.
- `n_iterations` -- Number of iterations of the correction for each x/y cycle.
- `prezero_correctors` -- Flag indicating whether to set the correctors to zero before starting.
- `track_before_and_after` -- Flag indicating whether tracking should be done both before and after correction.
- `start_from_centroid` -- Flag indicating that correction should start from the beam centroid. For orbit correction, only the beam momentum centroid is relevant.
- `use_actual_beam` -- Flag indicating that correction should employ tracking of the beam distribution rather than a single particle.
- `closed_orbit_accuracy` -- Accuracy of closed orbit computation.
- `closed_orbit_iterations` -- Number of iterations of closed orbit computation.
- `closed_orbit_iteration_fraction` -- Fraction of change in closed orbit to use at each iteration.

---

next up previous

Next: correction_matrix_output **Up:** Namelist Command Dictionary **Previous:** closed_orbit Michael Borland 2000-04-10

correction_matrix_output

- type: setup/action command.
- function: provide output of the orbit/trajectory correction matrix.

```
&correction_matrix_output
    STRING response[2] = NULL, NULL;
    STRING inverse[2] = NULL, NULL;
    long KnL_units = 0;
    long BnL_units = 0;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long fixed_length = 0;
&end
```

- `response` -- Array of (incomplete) filenames for SDDS output of the x and y response matrices. Recommnended values: ``
- `inverse` -- Array of (incomplete) filenames for SDDS output of the x and y inverse response matrices. Recommnended values: ``
- `KnL_units` -- Flag that, if set, indicates use of ``units'' of m/K0L rather than m/rad. This results in a sign change for the horizontal data.
- `BnL_units` -- Flag that, if set, indicates use of ``units'' of m/(T*m) rather than m/rad. This is useful for linac work in that the responses are automatically scaled with beam energy.
- `output_at_each_step` -- Flag that, if set, specifies output of the data at each simulation step. By default, the data is output immediately for the defined lattice.
- `output_before_tune_correction` -- Flag that, if set, specifies that when `output_at_each_step` is set, that output shall occur prior to correcting the tunes.
- `fixed_length` -- Flag that, if set, specifies output of the fixed-path-length matrix.

Michael Borland 2000-04-10

correct_tunes

- type: setup command.
- function: set up for correction of the tunes.

```
&correct_tunes
    STRING quadrupoles = NULL;
    double tune_x = 0;
    double tune_y = 0;
    long n_iterations = 5;
    double correction_fraction = 0.9;
    double tolerance = 0;
    long step_up_interval = 0;
    double max_correction_fraction = 0.9;
    double delta_correction_fraction = 0.1;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    long use_perturbed_matrix = 0;
&end
```

- `quadrupoles` -- List of names of quadrupoles to be used. Only two may be given.
- `tune_x`, `tune_y` -- Desired x and y tune values. If not given, the desired values are assumed to be the unperturbed tunes.
- `n_iterations` -- The number of iterations of the correction to perform.
- `correction_fraction` -- The fraction of the correction to apply at each iteration.
- `tolerance` -- When both tunes are within this value of the desired tunes, the iteration is stopped.
- `step_up_interval` -- Interval between increases in the correction fraction.
- `max_correction_fraction` -- Maximum correction fraction to allow.
- `delta_correction_fraction` -- Change in correction fraction after each `step_up_interval` steps.
- `strength_log` -- The (incomplete) name of a SDDS file to which the quadrupole strengths will be written as correction proceeds. Recommended value: ``%s.qst''.
- `change_defined_values` -- Changes the defined values of the quadrupole strengths. This means that when the lattice is saved (using `save_lattice`), the quadrupoles will have the corrected values. This would be used for correcting the tunes of a design lattice, for example, but not for correcting tunes of a perturbed lattice.
- `use_perturbed_matrix` -- If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain

correction. In general, it is not necessary and only slows the simulation.

---

next up previous

Next: error_element **Up:** Namelist Command Dictionary **Previous:** correction_matrix_output Michael Borland 2000-04-10

error_element

- type: setup command.
- function: assert a random error defintion for the accelerator.

```
&error_element
    STRING name = NULL;
    STRING element_type = NULL;
    STRING item = NULL;
    STRING type = "gaussian";
    double amplitude = 0.0;
    double cutoff = 3.0;
    long bind = 1;
    long bind_number = 0;
    long post_correction = 0;
    long fractional = 0;
    long additive = 1;
    STRING after = NULL;
    STRING before = NULL;
&end
```

- `name` -- The possibly wildcarded name of the elements for which errors are being specified.
- `element_type` -- An optional, possibly wildcarded string giving the type of elements to which the errors should be applied. E.g., `element_type=*MON*` would match all beam position monitors. If this item is given, then `name` may be left blank.
- `item` -- The parameter of the elements to which the error pertains.
- `type` -- The type of random distribution to use. May be one of ``uniform", ``gaussian", or ``plus_or_minus". A ``plus_or_minus" error is equal in magnitude to the amplitude given, with the sign randomly chosen.
- `amplitude` -- The amplitude of the errors.
- `cutoff` -- The cutoff for the random distribution.
- `bind`, `bind_number` -- If `bind_number` is positive, then a positive value of `bind` indicates that `bind_number` successive elements having the same name will have the same error value. If `bind` is negative, then the sign of the error is alternated between successive elements.
- `post_correction` -- A flag indicating whether the errors should be added after orbit, tune, and chromaticity correction.
- `fractional` -- A flag indicating whether the errors are fractional, in which case the amplitude refers to the amplitude of the fractional error.

- `additive` -- A flag indicating that the errors should be added to the prior value of the parameter. If zero, then the errors replace the prior value of the parameter.
- `after` -- The name of an element. If given, the error is applied only to elements that follow the named element in the beamline.
- `before` -- The name of an element. If given, the error is applied only to elements that precede the named element in the beamline.

---

next up previous

Next: error_control **Up:** Namelist Command Dictionary **Previous:** correct_tunes Michael Borland 2000-04-10

error_control

- type: setup command
- function: overall control of random errors.

```
&error_control
    long clear_error_settings = 1;
    long summarize_error_settings = 0;
    STRING error_log = NULL;
&end
```

- `clear_error_settings` -- Clear all previous error settings.
- `summarize_error_settings` -- Summarize current error settings.
- `error_log` -- The (incomplete) name of a SDDS file to which error values will be written. Recommended value: ``%s.erl''.

Michael Borland 2000-04-10

find_aperture

- type: action command.
- function: find the aperture in (x, y) space for an accelerator.

```
&find_aperture
    STRING output = NULL;
    STRING boundary = NULL;
    STRING mode = "many-particle";
    double xmin = -0.1;
    double xmax =  0.1;
    double ymin =  0.0;
    double ymax =  0.1;
    long nx  = 21;
    long ny  = 11;
    long n_splits = 0;
    double split_fraction = 0.5;
    double desired_resolution = 0.01;
    long verbosity = 0;
    long assume_nonincreasing = 0;
&end
```

- `output` -- The (incomplete) name of an SDDS file to send output to. Recommended value: ``%s.aper''.
- `boundary` -- The (incomplete) name of an SDDS file for the boundary points of the aperture search. Recommended value: ``%s.bnd''.
- `xmin`, `xmax`, `ymin`, `ymax` -- Region of the aperture search.
- `mode` -- May be ``many-particle'' or ``single-particle''. Many-particle searching is much faster, but does not allow interval splitting to search for the aperture boundary.
- `nx` -- Number of x values to take in initial search.
- `ny` -- Number of y values to take in search.
- `n_splits` -- If positive, the number of times to do interval splitting. Interval splitting refers to searching between the original grid points in order to refine the results.
- `split_fraction` -- If interval splitting is done, how the interval is split.
- `desired_resolution` -- If interval splitting is done, fraction of `xmax-xmin` to which to resolve the aperture.
- `assume_nonincreasing` -- If this variable is non-zero, the search assumes that the aperture at

$y + \mathrm{sign}(y) * \Delta y$ is no larger than that at y. This results in tracking of fewer particles but may give a pessimistic result.

- `verbosity` -- A larger value results in more printouts during computations.

---

next up previous

Next: floor_coordinates **Up:** Namelist Command Dictionary **Previous:** error_control Michael Borland 2000-04-10

floor_coordinates

- type: action command.
- function: compute floor coordinates for an accelerator.

```
&floor_coordinates
    STRING filename = NULL;
    double X0 = 0.0;
    double Z0 = 0.0;
    double theta0 = 0.0;
    long include_vertices = 0;
    long vertices_only = 0;
    long magnet_centers = 0;
&end
```

- `filename` -- The (incomplete) name of an SDDS file to send output to. Recommended value: ``%s.flr''.
- `X0`, `Z0`, `theta0` -- Initial X, Z, and angle coordinate of the
- `include_vertices` -- Flag that, if set, specifies including in the output the coordinates of the vertices of bending magnets.
- `vertices_only` -- Flag that, if set, specifies that output will contain only the coordinates of the vertices of bending magnets.
- `magnet_centers` -- Flag that, if set, specifies that output will contain the coordinates of the centers of all magnets. By default, the coordinates of the downstream end are given.

Michael Borland 2000-04-10

link_control

- type: setup command.
- function: overall control of element parameter links.

```
&link_control
    long clear_links = 1;
    long summarize_links = 0;
    long verbosity = 0;
&end
```

- `clear_links` -- Clear all previously set links.
- `summarize_links` -- Summarize all current set links.
- `verbosity` -- A larger value results in more output during computations.

Michael Borland 2000-04-10

next up previous

Next: load_parameters **Up:** Namelist Command Dictionary **Previous:** link_control

link_elements

- type: setup command.
- function: assert a link between parameters of accelerator elements.

```
&link_elements
    STRING target = NULL;
    STRING exclude = NULL;
    STRING item = NULL;
    STRING source = NULL;
    STRING source_position = "before";
    STRING mode = "dynamic";
    STRING equation = NULL;
&end
```

- `target` -- The name of the elements to be modified by the link. May contain wild-cards.
- `exclude` -- Wildcard sequence to match to element names. If a match is found, the element is excluded from the link.
- `item` -- The parameter that will be modified.
- `source` -- The name of the elements to be linked to.
- `source_position` -- May be one of ``before", ``after", ``adjacent", ``nearest", or ``same-occurrence".
- `mode` -- May be either ``dynamic" or ``static". A dynamic link is asserted whenever the source is changed (during correction, for example). A static link is asserted only when an error or variation is imparted to the source, and at the end of correction.
- `equation` -- An `rpn` equation for the new item value in terms of the item values for the source. The prior value of the item is on the top of the stack. To refer to the source parameter values, use the name of the parameters. These names must appear in capital letters.

Michael Borland 2000-04-10

load_parameters

- type: setup command.
- function: load parameters for elements from an SDDS file.

```
&load_parameters
        STRING filename = NULL;
        long change_defined_values = 0;
        long clear_settings = 0;
        long allow_missing_elements = 0;
        long verbose = 0;
&end
```

- `filename` -- Name of SDDS file from which to take data. The file must contain some of the following columns:
  - ElementName -- Required string column. The name of the element to change.
  - ElementParameter -- Required string column. The name of the parameter of the element to change.
  - ParameterValue -- Optional double column. If given, gives value of the parameter named in ElementParameter for element named in ElementName.
  - ParameterValueString -- Optional string column. If ParameterValue is not present, then this column must be present. The string data will be scanned, if necessary, to obtain a value for the parameter.
  - ParameterMode -- Optional string column. If given, for each row the value must be one of ``absolute", ``differential", ``ignore", or ``fractional".

  Unless `change_defined_values` is set, successive pages of the file are used for successive steps of the simulation. Several `elegant` commands generate output that may be used (on a subsequent run) with `load_parameters`; among these are the tune and chromaticity correction commands and the `run_setup` command (parameters output).
- `change_defined_values` -- Changes the defined values of the parameters. This means that when the lattice is saved (using `save_lattice`), the parameters will have the altered values. Also, if you want to alter the values for all steps of the simulation, you must set this flag.
- `clear_settings` -- If set, clear all settings and files being used for loading parameters.
- `allow_missing_elements` -- If set, allow elements in the file that are not in the lattice. In this case, the nonapplicable data is simply ignored.
- `verbose` -- If set, provide informational printouts about changes to parameters.

---

next up previous

Next: matrix_output **Up:** Namelist Command Dictionary **Previous:** link_elements Michael Borland 2000-04-10

matrix_output

- type: setup/action command.
- function: generate matrix output, or set up to do so later.

```
&matrix_output
    STRING printout = NULL;
    long printout_order = 1;
    long full_matrix_only = 0;
    STRING SDDS_output = NULL;
    long SDDS_output_order = 1;
    STRING SDDS_output_match = NULL;
    long output_at_each_step = 0;
    STRING start_from = NULL;
    long start_from_occurrence = 1;
&end
```

- `printout` -- The (incomplete) name of a file to which the matrix output will be printed (as text). Recommended value: ``%s.mpr''.
- `printout_order` -- The order to which the matrix is printed.
- `full_matrix_only` -- A flag indicating that only the matrix of the entire accelerator is to be output.
- `SDDS_output` -- The (incomplete) name of an SDDS file to which the matrix will be written. Recommended value: ``%s.mat''.
- `SDDS_output_order` -- The order to which the matrix is output in SDDS format.
- `SDDS_output_match` -- A wildcard string which element names must match in order for data to appear in the SDDS output file.
- `output_at_each_step` -- A flag indicating whether matrix output is desired at every simulation step.
- `start_from` -- The optional name of the accelerator element from which to begin concatenation and output.
- `start_from_occurrence` -- If `start_from` is not NULL, the number of the occurrence of the named element from which to start.

Michael Borland 2000-04-10

optimize

- type: action command.
- function: perform optimization.

```
&optimize
     long summarize_setup = 0;
&end
```

- `summarize_setup` -- A flag indicating, if set, that a summary of the optimization parameters should be printed.

Michael Borland 2000-04-10

optimization_constraint

- type: setup command.
- function: define a constraint for optimization.

```
&optimization_constraint
    STRING quantity = NULL;
    double lower = 0;
    double upper = 0;
&end
```

- `quantity` -- The quantity to be constrained, given as an `rpn` expression in terms of the optimization variables, the optimization covariables, and and the ``final'' parameters (see the entry for `run_setup` for the last of these). The optimization (co)variables are referred to as `<element-name>.<parameter-name>`, in all capital letters.
- `lower`, `upper` -- The lower and upper limits allowed for the expression.

Michael Borland 2000-04-10

optimization_covariable

- type: setup command.
- function: define an element parameter to be varied as a function of optimization parameters.

```
&optimization_covariable
    STRING name = NULL;
    STRING item = NULL;
    STRING equation = NULL;
&end
```

- `name` -- The name of the element.
- `item` -- The parameter of the element to be changed.
- `equation` -- An `rpn` equation for the value of the parameter in terms of the values of any parameters of any optimization variable. These latter appear in the equation in the form `<element-name>.<parameter-name>`, in all capital letters.

Michael Borland 2000-04-10

optimization_setup

- type: setup command.
- function: define overall optimization parameters and methods.

```
&optimization_setup
    STRING equation = NULL;
    STRING mode = "minimize";
    STRING method = "simplex";
    double tolerance = -0.01;
    double target = 0;
    long soft_failure = 1;
    long n_passes = 2;
    long n_evaluations = 500;
    STRING log_file = NULL;
    long output_sparsing_factor = 0;
&end
```

- `equation` -- An `rpn` equation for the optimization function, expressed in terms of any parameters of any optimization variables and the ``final'' parameters of the beam (as recorded in the `final` output file available in the `run_setup` namelist). The optimization variables or covariables may appear in the equation in the form `<element-name>.<parameter-name>`, all in capital letters. Data from MARK elements with FITPOINT=1 may be used via symbols of the form `<element-name>#<occurrence-number>.<parameter-name>`, where `<parameter-name>` can be a Twiss parameter name (if the twiss command was given), a floor coordinate name (if the floor command was given, or a beam-size or centroid name. The parameter names are the same as those used in the corresponding output files. Beam sizes and centroids are from tracking the particle distribution.

  If the twiss command was given, you may also refer to statistics of Twiss parameters in the form `<statistic>.<parameter-name>`, where `<statistic>` is either `min` or `max`. You may also use the symbols `nux`, `dnux/dp`, (and corresponding symbols for y), `alphac`, and `alphac2`. These are the tune, chromaticity, and first- and second- order momentum compaction factors. The final values of the Twiss parameters are referred to simply as `betax`, `etax`, etc.

  If the twiss command was given and radiation integral computation was requested, you may use `ex0` and `Sdelta0` for the equilibrium emittance and energy spread, plus `J<plane>` and `tau<plane>` for the damping partition and damping time, where `<plane>` is `x`, `y`, or `delta`.

  If the `floor_coordinates` command was given, you may use `X`, `Z`, and `theta` to refer to the final values of the floor coordinates.

Finally, you may use any of the names from the ``final'' output file (see `run_setup`), e.g., `Sx` (x beamsize) or `eny` (y normalized emittance). These refer to tracked properties of the beam.

The equation may be left blank, in which case the user must give one or more `optimization_term` commands. These use the same symbols, of course.

- `mode` -- May be either ``minimize'' or ``maximize''.
- `method` -- May be one of ``simplex'', ``grid'', and ``sample''.
- `tolerance` -- The convergence criterion for the optimization, with a negative value indicating a fractional criterion.
- `target` -- The value which, if reached, results in immediate termination of the optimization, whether it has converged or not.
- `soft_failure` -- A flag indicating, if set, that failure of an optimization pass should not result in termination of the optimization.
- `n_passes` -- The number of optimization passes made to achieve convergence (``simplex'' only).
- `n_evaluations` -- The number of allowed evaluations of the optimization function. If simplex optimization is used, this is the number of allowed evaluations per pass.
- `log_file` -- A file to which progress reports will be written as optimization proceeds. For SDDS data, use the `final` output file from the `run_setup` namelist.
- `output_sparsing_factor` -- If set to a value larger than 0, results in sparsing of output to the ``final'' file (see `run_setup`). This can make a significant difference in the optimization speed.

---

next up previous

Next: optimization_term **Up:** Namelist Command Dictionary **Previous:** optimization_covariable

Michael Borland 2000-04-10

optimization_term

- type: setup command.
- function: define optimization equation via individual terms

```
&optimization_
    STRING term = NULL;
&end
```

- `term` -- An `rpn` expression giving one term to be optimized. If more than one `optimization_term` command is given, then the terms are added. The advantage of using this command over giving an equation via `optimization_setup` is that `elegant` will report the value of each term as it performs the optimization (if a `log_file` is given to `optimization_setup`). This permits determination of which terms are causing problems for the optimization.

Michael Borland 2000-04-10

optimization_variable

- type: setup command.
- function: defines a parameter of an element to be used in optimization.

```
&optimization_variable
    STRING name = NULL;
    STRING item = NULL;
    double lower_limit = 0;
    double upper_limit = 0;
    double step_size = 1;
&end
```

- `name` -- The name of the element.
- `item` -- The parameter of the element to be varied.
- `lower_limit`, `upper_limit` -- The lower and upper limits allowed for the parameter. If these are equal, the range of the parameter is unlimited.
- `step_size` -- The initial step size (``simplex'' optimization ) or the grid size in this dimension (``grid'' or ``sample'' optimization).

Michael Borland 2000-04-10

print_dictionary

- type: action command.
- function: print dictionary of supported accelerator elements.

```
&print_dictionary
    STRING filename = NULL;
&end
```

- `filename` -- The name of a text file to which the dictionary will be printed.

Michael Borland 2000-04-10

rpn_expression

- type: action/setup command.
- function: pass an expression directly to the rpn submodule for execution.

```
&rpn_expression
    STRING expression = NULL;
&end
```

- `expression` -- An `rpn` expression. This expression is executed immediately and can be used, for example, to read in `rpn` commands from a file or store values in `rpn` memories.

Michael Borland 2000-04-10

run_control

- type: setup command.
- function: set up the number of simulation steps and passes.

```
&run_control
    long n_steps = 1;
    double bunch_frequency = 0;
    long n_indices = 0;
    long n_passes = 1;
&end
```

- `n_steps` -- The number of separate repetitions of the action implied by the next action command. If random errors are defined, this is also the number of separate error ensembles.
- `bunch_frequency` -- The frequency to use in calculating the time delay between repetitions.
- `n_indices` -- The number of looping indices for which to expect definitions in subsequent `vary_element` commands. If nonzero, then `n_steps` is ignored.
- `n_passes` -- The number of passes to make through the beamline per repetition.

Michael Borland 2000-04-10

run_setup

- type: setup command.
- function: set global parameters of the simulation and define primary input and output files.

```
&run_setup
    STRING lattice = NULL;
    STRING use_beamline = NULL;
    STRING rootname = NULL;
    STRING output = NULL;
    STRING centroid = NULL;
    STRING sigma = NULL;
    STRING final = NULL;
    STRING acceptance = NULL;
    STRING losses = NULL;
    STRING magnets = NULL;
    STRING semaphore_file = NULL;
    STRING parameters = NULL;
    long combine_bunch_statistics = 0;
    long wrap_around = 1;
    long default_order = 2;
    long concat_order = 0;
    long print_statistics = 0;
    long random_number_seed = 987654321;
    long correction_iterations = 1;
    double p_central = 0.0;
    STRING expand_for = NULL;
    long tracking_updates = 1;
    long echo_lattice = 0;
&end
```

- `lattice` -- Name of the lattice definition file.
- `echo_lattice` -- If nonzero, the lattice input is echoed to the standard output as the lattice is parsed. This can help detect certain problems with the lattice that cause `elegant` to crash.
- `use_beamline` -- Name of the beamline to use.
- `rootname` -- Filename fragment used in forming complete names from incomplete filenames. By default, the filename minus extension of the input file is used.
- `output` -- The (incomplete) name of an SDDS file into which final phase-space coordinates will be written. Recommended value: ``%s.out''.

- `centroid` -- The (incomplete) name of an SDDS file into which beam centroids as a function of s will be written. Recommended value: ``%s.cen".
- `sigma` -- The (incomplete) name of an SDDS file into which the beam sigma matrix as a function of z will be written. Recommended value: ``%s.sig".
- `final` -- The (incomplete) name of an SDDS file into which final beam and transport parameters will be written. Recommended value: ``%s.fin".
- `acceptance` -- The (incomplete) name of an SDDS file into which the initial coordinates of transmitted particles will be written. Recommended value: ``%s.acc".
- `losses` -- The (incomplete) name of an SDDS file into which information on lost particles will be written. Recommended value: ``%s.lost".
- `magnets` -- The (incomplete) name of an SDDS file into which a magnet layout representation will be written. Recommended value: ``%s.mag".
- `semaphore_file` -- The (incomplete) name of file that will be created just before exit from the program, but only if no errors occured. If the file exists, it is deleted. This file can be used to record the fact that the run completed without error.
- `parameters` -- The (incomplete) name of an SDDS file into which parameters of accelerator elements are written.
- `combine_bunch_statistics` -- A flag indicating whether to combine statistical information for all simulation steps. If non-zero, then the `sigma` and `centroid` data will be combined over all simulation steps.
- `wrap_around` -- A flag indicating whether the z coordinate should wrap-around or increase monotonically in multipass simulations.
- `default_order` -- The default order of transfer matrices used for elements having matrices.
- `concat_order` -- If non-zero, the order of matrix concatenation used.
- `print_statistics` -- A flag indicating whether to print information as each element is tracked.
- `random_number_seed` -- A seed for the random number generators. If zero, a seed will be generated from the system clock.
- `correction_iterations` -- Number of iterations of tune and chromaticity correction.
- `p_central` -- Central momentum of the beamline, about which expansions are done.
- `expand_for` -- Name of an SDDS file containing particle information, from which the central momentum will be set. The file contents are the same as required for `elegant` input with the `sdds_beam` namelist.
- `tracking_updates` -- A flag indicating whether to print summary information about tracking.

---

next up previous

Next: sasefel **Up:** Namelist Command Dictionary **Previous:** run_control Michael Borland 2000-04-10

sasefel

- type: setup/action command.
- function: set parameters for computation of SASE FEL gain and other properties.

```
&sasefel
    STRING output = NULL;
    STRING model = "Ming Xie";
    double beta = 0;
    double undulator_K = 3.1;
    double undulator_period = 0.033;
&end
```

- `output` -- The (incomplete) filename of an SDDS file to which output will be written.
- `model` -- The name of the FEL model used. At present, only one model is supported; the ``Ming-Xie" model is based on the simple parametrization M. Xie[12].
- `beta` -- The value of the beta function, in meters.
- `undulator_K` -- The K parameter of the undulator.
- `undulator_period` -- The undulator period, in meters.

Michael Borland 2000-04-10

save_lattice

- type: action command.
- function: save the current accelerator element and beamline definitions.

```
&save_lattice
    STRING filename = NULL;
&end
```

- `filename` -- The (incomplete) name of a file to which the element and beamline definitions will be written. Recommended value: ``%s.new''.

Michael Borland 2000-04-10

sdds_beam

- type: setup command.
- function: set up for tracking of particle coordinates stored in an SDDS file.

```
&sdds_beam
    STRING input = NULL;
    STRING input_type = "elegant";
    long n_particles_per_ring = 0;
    STRING selection_parameter = NULL;
    STRING selection_string = NULL;
    long one_random_bunch = 0;
    long reuse_bunch = 0;
    long prebunched = 0;
    long sample_interval = 1;
    long n_tables_to_skip = 0;
    long center_transversely = 0;
    long center_arrival_time = 0;
    double sample_fraction = 1;
    double p_lower = 0.0;
    double p_upper = 0.0;
    long save_initial_coordinates = 1;
&end
```

- `input` -- Name of an SDDS file containing coordinates of input particles.
- `input_type` -- May be ``elegant'' or ``spiffe'', indicating the name of the program that wrote the input file. The expected data quantities for the different types are:
  - `elegant`: $(\mathrm{x}, \mathrm{xp}, \mathrm{y}, \mathrm{yp}, \mathrm{t}, \mathrm{p})$, where x and y are in meters, $\mathrm{xp} = \mathrm{x}'$ and $\mathrm{xp} = \mathrm{y}'$ are dimensionless, t is in picoseconds, and $\mathrm{p} = \beta\gamma$ is the dimensionless momentum.
  - `spiffe`: $(\mathrm{r}, \mathrm{z}, \mathrm{pr}, \mathrm{pz}, \mathrm{t})$, where r and z are in meters, $\mathrm{pr} = \beta_\mathrm{r}\gamma$, $\mathrm{pz} = \beta_\mathrm{z}\gamma$, and t is in picoseconds.
- `n_particles_per_ring` -- For `spiffe` data, gives the number of particles to generate for each ring of charge.
- `selection_parameter` -- The name of a parameter in the SDDS file to be used for selection of pages of data.

- `selection_string` -- The value of the `selection_parameter` selection parameter required for a page to be used. E.g., if you have a file from the `shower` program containing positrons, electrons, and photons, you might want to select only the positrons.
- `one_random_bunch` -- A flag indicating whether, for `spiffe` data, a new random distribution should be calculated for each step of the simulation.
- `prebunched` -- A flag indicating, if zero, that the entire file is one ``bunch,'' and otherwise that each page in the file is a different bunch.
- `sample_interval` -- If non-zero, only every $\texttt{sample_interval}^{\text{th}}$ particle is used.
- `n_tables_to_skip` -- Number of SDDS pages to skip at the beginning of the file.
- `center_transversely` -- If non-zero, the transverse centroids of the distribution are made to be zero.
- `center_arrival_time` -- If non-zero, the mean arrival time of particles at the start of the accelerator is set to zero.
- `sample_fraction` -- If non-unity, the randomly selected fraction of the distribution to use.
- `p_lower`, `p_upper` -- If different, the lower and upper limit on $\beta\gamma$ of particles to use.
- `save_initial_coordinates` -- A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are reread from disk each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

---

next up previous

Next: subprocess **Up:** Namelist Command Dictionary **Previous:** save_lattice Michael Borland 2000-04-10

subprocess

- type: action command.
- function: execute a system command in a shell.

```
&subprocess
    STRING command = NULL;
&end
```

- `command` -- The text of the command to execute. The command may use the sequence ``

Michael Borland 2000-04-10

steering_element

- type: setup command.
- function: setup for use of a given parameter of a given element as a steering corrector.
- note: any use of this command disables the built-in definition of HKICK, VKICK, and HVKICK elements as steering elements.

```
&steering_element
    STRING name = NULL;
    STRING item = NULL;
    STRING plane = "h";
    double tweek = 1e-3;
    double limit = 0;
    STRING strength_log = NULL;
&end
```

- `name` -- The name of the element.
- `item` -- The parameter of the element to be varied.
- `plane` -- May be either ``h'' or ``v'', for horizontal or vertical correction.
- `tweek` -- The amount by which to change the item to compute the steering strength.
- `limit` -- The maximum allowed absolute value of the item.
- `strength_log` -- The (incomplete) name of a SDDS file to which the strengths of the item will be written as correction proceeds.

Michael Borland 2000-04-10

twiss_output

- type: action/setup command.
- function: compute and output Twiss parameters, or set up to do so.

```
&twiss_output
    STRING filename = NULL;
    long matched = 1;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long final_values_only = 0;
    long statistics = 0;
    long radiation_integrals = 0;
    long concat_order = 3;
    double beta_x = 1;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double beta_y = 1;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    STRING reference_file = NULL;
    STRING reference_element = NULL;
    long reference_element_occurrence = 0;
&end
```

- `filename` -- The (incomplete) name of an SDDS file to which the Twiss parameters will be written. Recommended value: ``%s.twi''.
- `matched` -- A flag indicating, if set, that the periodic or matched Twiss parameters should be found.
- `output_at_each_step` -- A flag indicating, if set, that output is desired at each step of the simulation.
- `output_before_tune_correction` -- A flag indicating, if set, that output is desired both before and after tune correction.
- `final_values_only` -- A flag indicating, if set, that only the final values of the Twiss parameters should be output, and not the parameters as a function of s.
- `statistics` -- A flag indicating, if set, that minimum, maximum, and average values of Twiss parameters should be computed and included in output.

- `radiation_integrals` -- A flag indicating, if set, that radiation integrals should be computed and included in output.
- `beta_X`, `alpha_X`, `eta_X`, `etap_X` -- If `matched` is zero, the initial values for the X plane.
- `concat_order` -- Order of matrix concatenation to use for determining matrix for computation of Twiss parameters. Using a lower order will result in inaccuracy for nonlinear lattices with orbits. However, if the orbit is zero, it is much faster to use `concat_order=1`.
- `reference_file` -- If given, the name of a file from which twiss parameter data will be taken to give the starting values. Ignored if `matched` is nonzero. The file should have the beta and alpha functions with the same names as the file created by this command.
- `reference_element` -- Element in `reference_file` at which to take the twiss parameter values. If not given, the values at the last element in `reference_file` are used.
- `reference_element_occurrence` -- Ignored if `reference_element` is not given. Otherwise, the occurence number of `reference_element` to use. If 0, the last occurence is used.

---

next up previous

Next: track **Up:** Namelist Command Dictionary **Previous:** steering_element Michael Borland 2000-04-10

track

- type: action command.
- function: track particles.

```
&track
    long center_on_orbit = 0;
    long center_momentum_also = 1;
    long soft_failure = 1;
    long use_linear_chromatic_matrix = 0;
    long longitudinal_ring_only = 0;
&end
```

- `center_on_orbit` -- A flag indicating whether to center the beam transverse coordinates on the closed orbit before tracking.
- `center_momentum_also` -- A flag indicating whether to center the momentum coordinate also.
- `soft_failure` -- If there is an error during tracking (e.g., a failure of orbit correction), continue to produce file output. This creates essentially empty slots in the files corresponding to the failed steps.
- `use_linear_chromatic_matrix` -- For each particle, a first-order matrix is computed for the particular momentum offset of the particle using the linear chromaticity and linear dependence of the beta functions on momentum.
- `longitudinal_ring_only` -- Tracks longitudinal coordinates only for a ring.

Michael Borland 2000-04-10

vary_element

- type: setup command.
- function: define an index and/or tie a parameter of an element to it.

```
&vary_element
    long index_number = 0;
    long index_limit = 0;
    STRING name = NULL;
    STRING item = NULL;
    double initial = 0;
    double final = 0;
    long differential = 0;
    long multiplicative = 0;
    long geometric = 0;
    STRING enumeration_file = NULL;
    STRING enumeration_column = NULL;
&end
```

- `index_number` -- A non-negative integer giving the number of the index.
- `index_limit` -- A positive integer giving the number of values the index will take. Must be given if this `index_number` has not been listed in a previous `vary_element` command, unless `enumeration_file` is given.
- `name` -- The name of an element.
- `item` -- The parameter of the element to vary.
- `initial`, `final` -- The initial and final values of the parameter.
- `enumeration_file` -- Name of an SDDS file giving values for the item.
- `enumeration_column` -- Column of the SDDS file giving the values.
- `differential` -- If nonzero, the initial and final values are taken as offsets from the predefined value of the parameter.
- `multiplicative` -- If nonzero, the initial and final values are taken as multipliers to be applied to the predefined value of the parameter in order to obtain the actual initial and final values.
- `geometric` -- If nonzero, then variation is geometric rather than arithmetic.

Michael Borland 2000-04-10

Accelerator and Element Description

As mentioned in the introduction, `elegant` uses a variant of the MAD input format for describing accelerators. With some exceptions, the accelerator description for one program can be read by the other with no modification. Among the differences:

- `elegant` does not support the use of MAD-style equations to compute the value of a quantity. The `link_element` namelist command can be used for this purpose, and is actually more flexible than the method used by MAD. Also, `rpn`-style equations may be given in double-quotes; these are evaluated once only when the lattice is parsed.
- `elegant` does not support substitution of parameters in beamline definitions.
- `elegant` contains many elements that MAD does not have, such as kick elements, wake fields, and numerically integrated elements.
- The length of an input line is not limited to 80 characters in `elegant`, as it is in MAD. However, for compatibility, any lattice created by `elegant` will conform to this limit.

`elegant`'s `print_dictionary` command allows the user to obtain a list of names and short descriptions of all accelerator elements recognized by the program, along with the names, units, types, and default values of all parameters of each element. The present output of this command is listed in the next section. The reader is referred to the MAD manual[2] for details on sign conventions for angles, focusing strength, and so forth.

Element Dictionary

Subsections

- ALPH
- BMAPXY
- BUMPER
- CENTER
- CEPL
- CHARGE
- CSBEND
- CSRCSBEND
- CSRDRIFT
- DRIF
- ECOL
- ELSE
- ENERGY
- FMULT
- HISTOGRAM
- HKICK
- HMON
- IBSCATTER
- KICKER
- KPOLY
- KQUAD
- KSBEND
- KSEXT
- MAGNIFY
- MALIGN

- MARK
- MATR
- MATTER
- MAXAMP
- MODRF
- MONI
- MULT
- NIBEND
- NISEPT
- OCTU
- PEPPOT
- PFILTER
- QUAD
- QUFRINGE
- RAMPP
- RAMPRF
- RBEN
- RCOL
- RECIRC
- RFCA
- RFCW
- RFDF
- RFMODE
- RFTM
- RMDF
- ROTATE
- SAMPLE
- SBEN
- SCATTER
- SCRAPER
- SEXT
- SOLE
- SREFFECTS

- STRAY
- TMCF
- TRCOUNT
- TRFMODE
- TRWAKE
- TUBEND
- TWLA
- TWMTA
- TWPL
- VKICK
- VMON
- WAKE
- WATCH
- ZLONGIT
- ZTRANSVERSE

Michael Borland 2000-04-10

ALPH

An alpha magnet implemented as a matrix, up to 3rd order. PART is used to split the magnet into halves. XSN and DPN allow momentum filtration at the midpoint.

Parameter Name	Units	Type	Default
XMAX	M	double	0.0
XS1	M	double	0.0
XS2	M	double	0.0
DP1		double	-1
DP2		double	1
XPUCK	M	double	-1
WIDTHPUCK	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
PART		long	0
ORDER		long	0

Michael Borland 2000-04-10

BMAPXY

A map of Bx and By vs x and y.

Parameter Name	Units	Type	Default
L	M	double	0.0
STRENGTH	$NULL$	double	0.0
ACCURACY	$NULL$	double	0.0
METHOD	$NULL$	STRING	NULL
FILENAME	$NULL$	STRING	NULL

Michael Borland 2000-04-10

BUMPER

A time-dependent uniform-field rectangular kicker magnet with no fringe effects. The waveform is in mpl format, with time in seconds and amplitude normalized to 1.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
TILT	RAD	double	0.0
TIME_OFFSET	S	double	0.0
PERIODIC		long	0
PHASE_REFERENCE		long	0
FIRE_ON_PASS		long	0
WAVEFORM		STRING	NULL
SPATIAL_DEPENDENCE		STRING	NULL

Michael Borland 2000-04-10

CENTER

An element that centers the beam transversely on the ideal trajectory.

Parameter Name	Units	Type	Default
X	M	long	1
XP		long	1
Y	M	long	1
YP		long	1
ONCE_ONLY		long	0

Michael Borland 2000-04-10

CEPL

A numerically-integrated linearly-ramped electric field deflector.

Parameter Name	Units	Type	Default
L	M	double	0.0
RAMP_TIME	S	double	1e-09
TIME_OFFSET	S	double	0.0
VOLTAGE	V	double	0.0
GAP	M	double	0.01
STATIC_VOLTAGE	V	double	0.0
TILT	RAD	double	0.0
ACCURACY		double	0.0001
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
PHASE_REFERENCE		long	0
N_STEPS		long	100
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median

Michael Borland 2000-04-10

CHARGE

An element to establish the total charge of a beam. Active on first pass only. If given, overrides all charge specifications on other elements.

Parameter Name	Units	Type	Default
TOTAL	C	double	0.0
PER_PARTICLE	C	double	0.0

Michael Borland 2000-04-10

CSBEND

A canonical kick sector dipole magnet. INTEGRATION_ORDER may be 2 or 4.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
K1	$1/M^2$	double	0.0
K2	$1/M^3$	double	0.0
K3	$1/M^3$	double	0.0
K4	$1/M^4$	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT	RAD	double	0.0
H1	$1/M$	double	0.0
H2	$1/M$	double	0.0
HGAP	M	double	0.0
FINT		double	0.5
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0

ETILT		double	0.0
N_KICKS		long	4
NONLINEAR		long	1
SYNCH_RAD		long	0
EDGE1_EFFECTS		long	1
EDGE2_EFFECTS		long	1
INTEGRATION_ORDER		long	2
EDGE1_KICK_LIMIT		double	-1
EDGE2_KICK_LIMIT		double	-1
KICK_LIMIT_SCALING		long	0
USE_BN		long	0
B1	$1/M$	double	0.0
B2	$1/M^2$	double	0.0
B3	$1/M^3$	double	0.0
B4	$1/M^4$	double	0.0

next up previous

Next: CSRCSBEND **Up:** Element Dictionary **Previous:** CHARGE Michael Borland 2000-04-10

CSRCSBEND

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
K1	$1/M^2$	double	0.0
K2	$1/M^3$	double	0.0
K3	$1/M^3$	double	0.0
K4	$1/M^4$	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT	RAD	double	0.0
H1	$1/M$	double	0.0
H2	$1/M$	double	0.0
HGAP	M	double	0.0
FINT		double	0.5
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0

ETILT		double	0.0
N_KICKS		long	4
NONLINEAR		long	1
SYNCH_RAD		long	0
EDGE1_EFFECTS		long	1
EDGE2_EFFECTS		long	1
INTEGRATION_ORDER		long	2
BINS		long	0
SG_HALFWIDTH		long	0
SG_ORDER		long	1
SGDERIV_HALFWIDTH		long	0
SGDERIV_ORDER		long	1
OUTPUT_FILE		STRING	NULL
OUTPUT_INTERVAL		long	1
STEADY_STATE		long	0
USE_BN		long	0
B1	$1/M$	double	0.0
B2	$1/M^2$	double	0.0
B3	$1/M^3$	double	0.0
B4	$1/M^4$	double	0.0

next up previous

Next: CSRDRIFT **Up:** Element Dictionary **Previous:** CSBEND Michael Borland 2000-04-10

CSRDRIFT

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.

Parameter Name	Units	Type	Default
L	M	double	0.0
ATTENUATION_LENGTH	M	double	0.0
N_KICKS		long	1
SPREAD		long	0
DZ		double	0.0

Michael Borland 2000-04-10

DRIF

A drift space implemented as a matrix, up to 2nd order

Parameter Name	Units	Type	Default
L	M	double	0.0
ORDER		long	`0`

Michael Borland 2000-04-10

ECOL

An elliptical collimator.

Parameter Name	Units	Type	Default
L	M	double	0.0
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

ELSE

Not implemented.

Parameter Name	Units	Type	Default

Michael Borland 2000-04-10

ENERGY

An element that matches the central momentum to the beam momentum, or changes the central momentum or energy to a specified value.

Parameter Name	Units	Type	Default
CENTRAL_ENERGY	MC^2	double	0.0
CENTRAL_MOMENTUM	MC	double	0.0
MATCH_BEAMLINE		long	0
MATCH_PARTICLES		long	0

Michael Borland 2000-04-10

FMULT

Multipole kick element with coefficient input from an SDDS file.

Parameter Name	Units	Type	Default
L	M	double	0.0
TILT	RAD	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0
N_KICKS		long	1
SYNCH_RAD		long	0
FILENAME		STRING	NULL

Michael Borland 2000-04-10

HISTOGRAM

Request for histograms of particle coordinates to be output to SDDS file.

Parameter Name	Units	Type	Default
FILENAME		STRING	
INTERVAL		long	1
START_PASS		long	0
BINS		long	50
FIXED_BIN_SIZE		long	0
X_DATA		long	1
Y_DATA		long	1
LONGIT_DATA		long	1
BIN_SIZE_FACTOR		double	1

Michael Borland 2000-04-10

HKICK

A horizontal steering dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
KICK	RAD	double	0.0
TILT	RAD	double	0.0
B2	$1/M^2$	double	0.0
CALIBRATION		double	1
EDGE_EFFECTS		long	0
ORDER		long	0
STEERING		long	1

Michael Borland 2000-04-10

HMON

A horizontal position monitor, accepting a rpn equation for the readout as a function of the actual position (x).

Parameter Name	Units	Type	Default
L	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
WEIGHT		double	1
TILT		double	0.0
CALIBRATION		double	1
ORDER		long	0
READOUT		STRING	NULL

Michael Borland 2000-04-10

IBSCATTER

A simulation of intra-beam scattering.

Parameter Name	Units	Type	Default
COUPLING		double	1
FRACTION		double	1
CHARGE	C	double	0.0

Michael Borland 2000-04-10

KICKER

A combined horizontal-vertical steering magnet implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
HKICK	RAD	double	0.0
VKICK	RAD	double	0.0
TILT	RAD	double	0.0
B2	$1/M^2$	double	0.0
HCALIBRATION		double	1
VCALIBRATION		double	1
EDGE_EFFECTS		long	0
ORDER		long	0
STEERING		long	1

Michael Borland 2000-04-10

KPOLY

A thin kick element with polynomial dependence on the coordinates in one plane.

Parameter Name	Units	Type	Default
COEFFICIENT	M^{-ORDER}	double	0.0
TILT	RAD	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FACTOR		double	1
ORDER		long	0
PLANE		STRING	x

Michael Borland 2000-04-10

KQUAD

A canonical kick quadrupole, which differs from the MULT element with ORDER=1 in that it can be used for tune correction.

Parameter Name	Units	Type	Default
L	M	double	0.0
K1	$1/M^2$	double	0.0
TILT	RAD	double	0.0
BORE	M	double	0.0
B	T	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE	M	double	0.0
N_KICKS		long	4
SYNCH_RAD		long	0
SYSTEMATIC_MULTIPOLES		STRING	NULL
RANDOM_MULTIPOLES		STRING	NULL
INTEGRATION_ORDER		long	4

Michael Borland 2000-04-10

KSBEND

A kick bending magnet which is NOT canonical, but is better than a 2nd order matrix implementation.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
K1	$1/M^2$	double	0.0
K2	$1/M^3$	double	0.0
K3	$1/M^3$	double	0.0
K4	$1/M^4$	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT	RAD	double	0.0
H1	$1/M$	double	0.0
H2	$1/M$	double	0.0
HGAP	M	double	0.0
FINT		double	0.5
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0

ETILT		double	0.0
N_KICKS		long	4
NONLINEAR		long	1
SYNCH_RAD		long	0
EDGE1_EFFECTS		long	1
EDGE2_EFFECTS		long	1
EDGE_ORDER		long	1
PARAXIAL		long	0
TRANSPORT		long	0
METHOD		STRING	modified-midpoint

next up previous

Next: KSEXT **Up:** Element Dictionary **Previous:** KQUAD Michael Borland 2000-04-10

KSEXT

A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.

Parameter Name	Units	Type	Default
L	M	double	0.0
K2	$1/M^3$	double	0.0
TILT	RAD	double	0.0
BORE	M	double	0.0
B	T	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE	M	double	0.0
N_KICKS		long	4
SYNCH_RAD		long	0
SYSTEMATIC_MULTIPOLES		STRING	NULL
RANDOM_MULTIPOLES		STRING	NULL
INTEGRATION_ORDER		long	4

Michael Borland 2000-04-10

MAGNIFY

An element that allows multiplication of phase-space coordinates of all particles by constants.

Parameter Name	Units	Type	Default
MX		double	1
MXP		double	1
MY		double	1
MYP		double	1
MS		double	1
MDP		double	1

Michael Borland 2000-04-10

MALIGN

A misalignment of the beam, implemented as a zero-order matrix.

Parameter Name	Units	Type	Default
DXP		double	0.0
DYP		double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
DT	S	double	0.0
DP		double	0.0
DE		double	0.0
ON_PASS		long	-1

Michael Borland 2000-04-10

MARK

A marker, equivalent to a zero-length drift space.

Parameter Name	Units	Type	Default
FITPOINT		long	0

Michael Borland 2000-04-10

MATR

Explicit matrix input from a text file, in the format written by the print_matrix command.

Parameter Name	Units	Type	Default
L	M	double	0.0
FILENAME		STRING	
ORDER		long	1

Michael Borland 2000-04-10

MATTER

A Coulomb-scattering and energy-absorbing element simulating material in the beam path.

Parameter Name	Units	Type	Default
L	M	double	0.0
XO	M	double	0.0
ELASTIC		long	0

Michael Borland 2000-04-10

MAXAMP

A collimating element that sets the maximum transmitted particle amplitudes for all following elements, until the next MAXAMP.

Parameter Name	Units	Type	Default
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
ELLIPTICAL		long	0

Michael Borland 2000-04-10

MODRF

A first-order matrix RF cavity with exact phase dependence, plus optional amplitude and phase modulation.

Parameter Name	Units	Type	Default
L	M	double	0.0
VOLT	V	double	0.0
PHASE	DEG	double	0.0
FREQ	Hz	double	500000000
Q		double	0.0
PHASE_REFERENCE		long	`0`
AMMAG		double	0.0
AMPHASE	DEG	double	0.0
AMFREQ	Hz	double	0.0
PMMAG	DEG	double	0.0
PMPHASE	DEG	double	0.0
PMFREQ	Hz	double	0.0
FIDUCIAL		STRING	NULL

Michael Borland 2000-04-10

MONI

A two-plane position monitor, accepting two rpn equations for the readouts as a function of the actual positions (x and y).

Parameter Name	Units	Type	Default
L	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
WEIGHT		double	1
TILT		double	0.0
XCALIBRATION		double	1
YCALIBRATION		double	1
ORDER		long	0
XREADOUT		STRING	NULL
YREADOUT		STRING	NULL

Michael Borland 2000-04-10

MULT

A canonical kick multipole.

Parameter Name	Units	Type	Default
L	M	double	0.0
KNL	$M^{(1-ORDER)}$	double	0.0
TILT	RAD	double	0.0
BORE	M	double	0.0
BNL	TM	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FACTOR		double	1
ORDER		long	1
N_KICKS		long	4
SYNCH_RAD		long	0

Michael Borland 2000-04-10

NIBEND

A numerically-integrated dipole magnet with various extended-fringe-field models.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT		double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FINT		double	0.5
HGAP	M	double	0.0
FP1	M	double	10
FP2	M	double	1
FSE		double	0.0
ETILT		double	0.0
ACCURACY		double	0.0001
MODEL		STRING	linear
METHOD		STRING	runge-kutta
SYNCH_RAD		long	0

Michael Borland 2000-04-10

NISEPT

A numerically-integrated dipole magnet with a Cartesian gradient.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
E1	RAD	double	0.0
B1	M	double	0.0
Q1REF	M	double	0.0
FLEN	M	double	0.0
ACCURACY		double	0.0001
METHOD		STRING	runge-kutta
MODEL		STRING	linear

Michael Borland 2000-04-10

OCTU

Not implemented-use the MULT element.

Parameter Name	Units	Type	Default

Michael Borland 2000-04-10

PEPPOT

A pepper-pot plate.

Parameter Name	Units	Type	Default
L	M	double	0.0
RADII	M	double	0.0
TRANSMISSION		double	0.0
TILT	RAD	double	0.0
THETA_RMS	RAD	double	0.0
N_HOLES		long	0

Michael Borland 2000-04-10

PFILTER

An element for energy and momentum filtration.

Parameter Name	Units	Type	Default
DELTALIMIT		double	-1
LOWERFRACTION		double	0.0
UPPERFRACTION		double	0.0
FIXPLIMITS		long	0

Michael Borland 2000-04-10

QUAD

A quadrupole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
K1	$1/M^2$	double	0.0
TILT	RAD	double	0.0
FFRINGE		double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE	M	double	0.0
ORDER		long	`0`

Michael Borland 2000-04-10

QUFRINGE

An element consisting of a linearly increasing or decreasing quadrupole field.

Parameter Name	Units	Type	Default
L	M	double	0.0
K1	$1/M^2$	double	0.0
TILT	RAD	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE	M	double	0.0
DIRECTION		long	0
ORDER		long	0

Michael Borland 2000-04-10

RAMPP

A momentum-ramping element that changes the central momentum according to a mpl format file of the momentum factor vs time in seconds.

Parameter Name	Units	Type	Default
WAVEFORM		STRING	NULL

Michael Borland 2000-04-10

RAMPRF

A voltage-ramped RF cavity, implemented like RFCA. The voltage ramp pattern is given by a mpl-format file of the voltage factor vs time in seconds.

Parameter Name	Units	Type	Default
L	M	double	0.0
VOLT	V	double	0.0
PHASE	DEG	double	0.0
FREQ	Hz	double	500000000
PHASE_REFERENCE		long	0
VOLT_WAVEFORM		STRING	NULL
PHASE_WAVEFORM		STRING	NULL
FREQ_WAVEFORM		STRING	NULL
FIDUCIAL		STRING	NULL

Michael Borland 2000-04-10

RBEN

A rectangular dipole, implemented as a SBEND with edge angles.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
K1	$1/M^2$	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT	RAD	double	0.0
K2	$1/M^3$	double	0.0
H1	$1/M$	double	0.0
H2	$1/M$	double	0.0
HGAP	M	double	0.0
FINT		double	0.5
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0
ETILT		double	0.0
EDGE1_EFFECTS		long	1
EDGE2_EFFECTS		long	1
ORDER		long	0

EDGE_ORDER		long	0
TRANSPORT		long	0
USE_BN		long	0
B1	$1/M$	double	0.0
B2	$1/M^2$	double	0.0

next up previous

Next: RCOL **Up:** Element Dictionary **Previous:** RAMPRF Michael Borland 2000-04-10

RCOL

A rectangular collimator.

Parameter Name	Units	Type	Default
L	M	double	0.0
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

RECIRC

An element that defines the point to which particles recirculate in multi-pass tracking

Parameter Name	Units	Type	Default
I_RECIRC_ELEMENT		long	0

Michael Borland 2000-04-10

RFCA

A first-order matrix RF cavity with exact phase dependence.

Parameter Name	Units	Type	Default
L	M	double	0.0
VOLT	V	double	0.0
PHASE	DEG	double	0.0
FREQ	Hz	double	500000000
Q		double	0.0
PHASE_REFERENCE		long	0
CHANGE_P0		long	0
CHANGE_T		long	0
FIDUCIAL		STRING	NULL
END1_FOCUS		long	0
END2_FOCUS		long	0
N_KICKS		long	1
DX	M	double	0.0
DY	M	double	0.0
T_REFERENCE	S	double	-1

Michael Borland 2000-04-10

RFCW

A combination of RFCA, WAKE, and TRWAKE.

Parameter Name	Units	Type	Default
L	M	double	0.0
CELL_LENGTH	M	double	0.0
VOLT	V	double	0.0
PHASE	DEG	double	0.0
FREQ	Hz	double	500000000
Q		double	0.0
PHASE_REFERENCE		long	0
CHANGE_P0		long	0
CHANGE_T		long	0
FIDUCIAL		STRING	NULL
END1_FOCUS		long	0
END2_FOCUS		long	0
N_KICKS		long	1
WAKEFILE		STRING	NULL
ZWAKEFILE		STRING	NULL
TRWAKEFILE		STRING	NULL
TCOLUMN		STRING	NULL
WXCOLUMN		STRING	NULL
WYCOLUMN		STRING	NULL
WZCOLUMN		STRING	NULL
N_BINS		long	0
INTERPOLATE		long	0

SMOOTHING		long	0
SG_HALFWIDTH		long	4
SG_ORDER		long	2
DX	M	double	0.0
DY	M	double	0.0

---

next up previous

Next: RFDF **Up:** Element Dictionary **Previous:** RFCA Michael Borland 2000-04-10

RFDF

A deflecting TM RF cavity, using an approximate analytical solution.

Parameter Name	Units	Type	Default
L	M	double	0.0
PHASE	DEG	double	0.0
TILT	RAD	double	0.0
FREQUENCY	HZ	double	2856000000
VOLTAGE	V	double	0.0
GAP	M	double	0.01
TIME_OFFSET	S	double	0.0
B_FIELD	T	double	0.0
N_SECTIONS		long	10
PHASE_REFERENCE		long	`0`
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

RFMODE

A simulation of a beam-driven TM monopole mode of a RF cavity.

Parameter Name	Units	Type	Default
RA	Ohm	double	0.0
RS	Ohm	double	0.0
Q		double	0.0
FREQ	Hz	double	0.0
CHARGE	C	double	0.0
INITIAL_V	V	double	0.0
INITIAL_PHASE	RAD	double	0.0
INITIAL_T	S	double	0.0
BETA		double	0.0
BIN_SIZE	S	double	0.0
N_BINS		long	20
PRELOAD		long	0
PRELOAD_FACTOR		double	1
RIGID_UNTIL_PASS		long	0
SAMPLE_INTERVAL		long	1
RECORD		STRING	NULL
SINGLE_PASS		long	0
PASS_INTERVAL		long	1

RFTM

Not implemented.

Parameter Name	Units	Type	Default
L	M	double	0.0
FREQUENCY	HZ	double	2856000000
PHASE	RAD	double	0.0
EZ_PEAK	V	double	0.0
TIME_OFFSET	S	double	0.0
RADIAL_OFFSET	M	double	0.0
TILT	RAD	double	0.0
ACCURACY		double	0.0001
PHASE_REFERENCE		long	0
N_STEPS		long	100
FILENAME		STRING	
DX	M	double	0.0
DY	M	double	0.0
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median

Michael Borland 2000-04-10

RMDF

A linearly-ramped electric field deflector, using an approximate analytical solution.

Parameter Name	Units	Type	Default
L	M	double	0.0
TILT	RAD	double	0.0
RAMP_TIME	S	double	1e-09
VOLTAGE	V	double	0.0
GAP	M	double	0.01
TIME_OFFSET	S	double	0.0
N_SECTIONS		long	10
PHASE_REFERENCE		long	0
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

ROTATE

An element that rotates the beam coordinates about the longitudinal axis.

Parameter Name	Units	Type	Default
TILT	RAD	double	0.0

Michael Borland 2000-04-10

SAMPLE

An element that reduces the number of particles in the beam by interval-based or random sampling.

Parameter Name	Units	Type	Default
FRACTION		double	1
INTERVAL		long	1

Michael Borland 2000-04-10

SBEN

A sector dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
K1	$1/M^2$	double	0.0
E1	RAD	double	0.0
E2	RAD	double	0.0
TILT	RAD	double	0.0
K2	$1/M^3$	double	0.0
H1	$1/M$	double	0.0
H2	$1/M$	double	0.0
HGAP	M	double	0.0
FINT		double	0.5
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE		double	0.0
ETILT		double	0.0
EDGE1_EFFECTS		long	1
EDGE2_EFFECTS		long	1
ORDER		long	0

EDGE_ORDER		long	0
TRANSPORT		long	0
USE_BN		long	0
B1	$1/M$	double	0.0
B2	$1/M^2$	double	0.0

---

next up previous

Next: SCATTER **Up:** Element Dictionary **Previous:** SAMPLE Michael Borland 2000-04-10

SCATTER

A scattering element to add gaussian random numbers to particle coordinates.

Parameter Name	Units	Type	Default
X	M	double	0.0
XP	M	double	0.0
Y	M	double	0.0
YP	M	double	0.0
DP	M	double	0.0

Michael Borland 2000-04-10

SCRAPER

A collimating element that sticks into the beam from one side only. The directions 0, 1, 2, and 3 are from +x, +y, -x, and -y, respectively.

Parameter Name	Units	Type	Default
L	M	double	0.0
POSITION	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
XO	M	double	0.0
INSERT_FROM		STRING	NULL
ELASTIC		long	0
DIRECTION		long	-1

Michael Borland 2000-04-10

SEXT

A sextupole implemented as a matrix, up to 2nd order

Parameter Name	Units	Type	Default
L	M	double	0.0
K2	$1/M^3$	double	0.0
TILT	RAD	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
FSE	M	double	0.0
ORDER		long	0

Michael Borland 2000-04-10

SOLE

A solenoid implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
KS	RAD/M	double	0.0
B	T	double	0.0
DX	M	double	0.0
DY	M	double	0.0
DZ	M	double	0.0
ORDER		long	0

Michael Borland 2000-04-10

SREFFECTS

Simulation of synchrotron radiation effects (damping and quantum excitation).

Parameter Name	Units	Type	Default
JX		double	1
JY		double	1
JDELTA		double	2
EXREF	m	double	0.0
EYREF	m	double	0.0
SDELTAREF	m	double	0.0
DDELTAREF		double	0.0
PREF	$m_e c$	double	0.0
COUPLING		double	0.0
FRACTION		double	1
DAMPING		long	1
QEXCITATION		long	1
LOSSES		long	1

Michael Borland 2000-04-10

STRAY

A stray field element with local and global components. Global components are defined relative to the initial beamline direction.

Parameter Name	Units	Type	Default
L	M	double	0.0
LBX	T	double	0.0
LBY	T	double	0.0
GBX	T	double	0.0
GBY	T	double	0.0
GBZ	T	double	0.0
ORDER		long	`0`

Michael Borland 2000-04-10

TMCF

A numerically-integrated accelerating TM RF cavity with spatially-constant fields.

Parameter Name	Units	Type	Default
L	M	double	0.0
FREQUENCY	HZ	double	2856000000
PHASE	S	double	0.0
TIME_OFFSET	S	double	0.0
RADIAL_OFFSET	M	double	1
TILT	RAD	double	0.0
ER	V	double	0.0
BPHI	T	double	0.0
EZ	V	double	0.0
ACCURACY		double	0.0001
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
PHASE_REFERENCE		long	0
N_STEPS		long	100
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median

Michael Borland 2000-04-10

TRCOUNT

An element that defines the point from which transmission calculations are made.

Parameter Name	Units	Type	Default
DUMMY		long	0

Michael Borland 2000-04-10

TRFMODE

A simulation of a beam-driven TM dipole mode of a RF cavity.

Parameter Name	Units	Type	Default
RA	Ohm	double	0.0
RS	Ohm	double	0.0
Q		double	0.0
FREQ	Hz	double	0.0
CHARGE	C	double	0.0
BETA		double	0.0
BIN_SIZE	S	double	0.0
N_BINS		long	20
PLANE		STRING	both
SINGLE_PASS		long	`0`
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

TRWAKE

Transverse wake specified as a function of time lag behind the particle.

Parameter Name	Units	Type	Default
INPUTFILE		STRING	NULL
TCOLUMN		STRING	NULL
WXCOLUMN		STRING	NULL
WYCOLUMN		STRING	NULL
CHARGE	C	double	0.0
FACTOR	C	double	1
N_BINS		long	128
INTERPOLATE		long	0
SMOOTHING		long	0
SG_HALFWIDTH		long	4
SG_ORDER		long	2
DX	M	double	0.0
DY	M	double	0.0

Michael Borland 2000-04-10

TUBEND

A special rectangular bend element for top-up backtracking.

Parameter Name	Units	Type	Default
L	M	double	0.0
ANGLE	RAD	double	0.0
FSE		double	0.0
OFFSET		double	0.0
MAGNET_WIDTH		double	0.0
MAGNET_ANGLE		double	0.0

Michael Borland 2000-04-10

TWLA

A numerically-integrated first-space-harmonic traveling-wave linear accelerator.

Parameter Name	Units	Type	Default
L	M	double	0.0
FREQUENCY	HZ	double	2856000000
PHASE	RAD	double	0.0
TIME_OFFSET	S	double	0.0
EZ	V/M	double	0.0
B_SOLENOID	T	double	0.0
ACCURACY		double	0.0001
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
BETA_WAVE		double	1
ALPHA	$1/M$	double	0.0
PHASE_REFERENCE		long	0
N_STEPS		long	100
FOCUSSING		long	1
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median
CHANGE_P0		long	0

Next: TWMTA **Up:** Element Dictionary **Previous:** TUBEND Michael Borland 2000-04-10

TWMTA

A numerically-integrated traveling-wave muffin-tin accelerator.

Parameter Name	Units	Type	Default
L	M	double	0.0
FREQUENCY	HZ	double	2856000000
PHASE	RAD	double	0.0
EZ	V/M	double	0.0
ACCURACY		double	0.0001
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
KX	$1/M$	double	0.0
BETA_WAVE		double	1
BSOL		double	0.0
ALPHA	$1/M$	double	0.0
PHASE_REFERENCE		long	0
N_STEPS		long	100
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median

Michael Borland 2000-04-10

TWPL

A numerically-integrated traveling-wave stripline deflector.

Parameter Name	Units	Type	Default
L	M	double	0.0
RAMP_TIME	S	double	1e-09
TIME_OFFSET	S	double	0.0
VOLTAGE	V	double	0.0
GAP	M	double	0.01
STATIC_VOLTAGE	V	double	0.0
TILT	RAD	double	0.0
ACCURACY		double	0.0001
X_MAX	M	double	0.0
Y_MAX	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
PHASE_REFERENCE		long	0
N_STEPS		long	100
METHOD		STRING	runge-kutta
FIDUCIAL		STRING	t,median

Michael Borland 2000-04-10

VKICK

A vertical steering dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default
L	M	double	0.0
KICK	RAD	double	0.0
TILT	RAD	double	0.0
B2	$1/M^2$	double	0.0
CALIBRATION		double	1
EDGE_EFFECTS		long	0
ORDER		long	0
STEERING		long	1

Michael Borland 2000-04-10

VMON

A vertical position monitor, accepting a rpn equation for the readout as a function of the actual position (y).

Parameter Name	Units	Type	Default
L	M	double	0.0
DX	M	double	0.0
DY	M	double	0.0
WEIGHT		double	1
TILT		double	0.0
CALIBRATION		double	1
ORDER		long	0
READOUT		STRING	NULL

Michael Borland 2000-04-10

WAKE

Longitudinal wake specified as a function of time lag behind the particle.

Parameter Name	Units	Type	Default
INPUTFILE		STRING	NULL
TCOLUMN		STRING	NULL
WCOLUMN		STRING	NULL
CHARGE	C	double	0.0
FACTOR	C	double	1
N_BINS		long	128
INTERPOLATE		long	0
SMOOTHING		long	0
SG_HALFWIDTH		long	4
SG_ORDER		long	2
CHANGE_P0		long	0

Michael Borland 2000-04-10

WATCH

A beam property/motion monitor-allowed modes are centroid, parameter, coordinate, and fft.

Parameter Name	Units	Type	Default
FRACTION		double	1
INTERVAL		long	1
START_PASS		long	0
FILENAME		STRING	
LABEL		STRING	
MODE		STRING	coordinates
X_DATA		long	1
Y_DATA		long	1
LONGIT_DATA		long	1

Michael Borland 2000-04-10

ZLONGIT

A simulation of a single-pass broad-band or functionally specified longitudinal impedance.

Parameter Name	Units	Type	Default
CHARGE	C	double	0.0
BROAD_BAND		long	0
RA	Ohm	double	0.0
RS	Ohm	double	0.0
Q		double	0.0
FREQ	Hz	double	0.0
ZREAL		STRING	NULL
ZIMAG		STRING	NULL
BIN_SIZE	S	double	0.0
N_BINS		long	128
WAKES		STRING	NULL
WAKE_INTERVAL		long	1
AREA_WEIGHT		long	0
INTERPOLATE		long	0
SMOOTHING		long	0
SG_ORDER		long	2
SG_HALFWIDTH		long	4

Michael Borland 2000-04-10

ZTRANSVERSE

A simulation of a single-pass broad-band or functionally-specified transverse dipole impedance.

Parameter Name	Units	Type	Default
CHARGE	C	double	0.0
BROAD_BAND		long	0
RS	Ohm	double	0.0
Q		double	0.0
FREQ	Hz	double	0.0
INPUTFILE		STRING	NULL
FREQCOLUMN		STRING	NULL
ZXREAL		STRING	NULL
ZXIMAG		STRING	NULL
ZYREAL		STRING	NULL
ZYIMAG		STRING	NULL
BIN_SIZE	S	double	0.0
INTERPOLATE		long	0
N_BINS		long	128
SMOOTHING		long	0
SG_ORDER		long	2
SG_HALFWIDTH		long	4
DX	M	double	0.0
DY	M	double	0.0

Examples

Example runs and post-processing files are included along with the distribution of `elegant`. These are drawn from the author's research and all concern various aspects of the Argonne Positron Accumulator Ring (PAR) and its injection and ejection lines (LTP and PTB, respectively).

The examples are intended to demonstrate program capabilities with minimal work on the user's part. Each demo is invoked using a command (a C-shell script) that can both run `elegant` and post-process the output. After running the demo, the output can be viewed again without rerunning `elegant` by invoking the command with the word `review` added to the command line. Including the word `hardcopy` on the command line results in the graphs being sent to your default printer, which is assumed to accept Postscript.

The post-processing is typically handled by a lower-level script that is called from the demo script. These lower-level scripts are good models for the creation of customized scripts for user applications.

1. `par10h*` -- These files provide a demonstration of Twiss parameter computation, tracking, element variation, and map analysis. The lattice is defined with kick elements, which are used for all tracking. After computation of the Twiss parameters for the PAR[6], a series of particles are tracked with different initial x coordinates. Finally, the tunes and Twiss parameters are computed by tracking; they are very close to the analytical values. The post-processing commands make phase-space plots and plots of FFTs of the motion, showing that the motion becomes chaotic at the stability limit. To execute this demo, type the command `par10h`.
2. `par_sympl*` -- These files provide a demonstration of the symplecticity of tracking with `elegant` kick elements. A single large-amplitude particle is tracked for 2^{14} turns. The invariant J_x is then computed and plotted as a function of turn number. To execute this demo, type the command `par_sympl`. The post-processing takes quite some time because of the very large number of points.
3. `par_chrom*` -- These files provide a demonstration of computing chromaticity and other parameters as a function of momentum offset using map analysis. The lattice is the same as `par10h.lte`, except all of the elements are implemented using second-order matrices. Hence, the chromaticity from tracking should be nearly identical to the analytical results computed by the `twiss_output` command, which it is. To run this demonstration, enter `par_chrom`. The reader may wish to try this demo again using `ksbend`, `csbend`, or `nibend` elements in place of the `sbend` elements, and `kquad` (`ksext`) elements in place of the `quad` (`sext`) elements.
4. `par_damp*` -- These files provide a demonstration of damping partition calculation using single turn tracking with synchrotron radiation. The expected value of the longitudinal damping partition

for PAR is $J_\delta = 1.758$. The user may edit the lattice file, `par_damp.lte`, to invoke a different element for the dipole magnet. In particular, definitions for numerically integrated dipoles with extended fringe-fields are present. To execute this demo, type the command `par_damp`.

5. `par_dynap*` -- These files provide a demonstration of dynamic aperture runs for a series of randomized machines. Also exhibited here are orbit, tune, and chromaticity correction. The post-processing commands make a plot of the dynamic apertures with the physical aperture superimposed. (The `orbcorr_plots` script can also be used to plot orbit correction information.) To execute this demo, type the command `par_dynap`. The lattice has been stripped down so that only a few of the more significant multipoles are present. Also, fictitious extra sextupoles have been added to compensate the lack of second-order edge terms in the bending magnets (these would result in nonsymplectic tracking if included). Still, the running time is many hours.
6. `ejoptk*` -- These files provide a demonstration of the optimization of a multi-turn ejection bump for PAR, using a time-dependent kicker waveform (formed from two cubic splines). After optimization, the lattice is tracked with a realistic beam distribution to verify good transmission and show the centroid position vs z over three turns. To execute this demo, type the command `ejoptk`.
7. `ltp_te*` -- These files provide a demonstration of transport line simulation. The Linac-to-PAR transport line is simulated with errors and trajectory correction to predict the transmission losses and the steering error at the exit of the septum. The trajectory correction uses tracking of a beam distribution, which is slower than tracking the centroid, but which produces better results in the presence of the large momentum spread. The reader may wish to verify this by turning off this feature and running the simulation again. To execute this demo, type the command `ltp_te`. The running time for this demo is quite long.

---

next up previous

Next: The rpn Calculator **Up:** User's Manual for elegant **Previous:** ZTRANSVERSE Michael Borland 2000-04-10

The rpn Calculator

The program rpn is a Reverse Polish Notation programmable scientific calculator written in C. It is incorporated as a subprogram into elegant, and a number of the SDDS programs. It also exists as a command-line program, rpnl, which executes its command-line arguments as rpn operations and prints the result before exiting. Use of rpn in any of these modes is extremely straightforward. Use of the program in its stand-alone form is the best way to gain familiarity with it. Once you've entered rpn, entering ``help'' will produce a list of the available operators with brief summaries of their function. Also, the rpn definitions file rpn.defns, distributed with elegant, gives examples of most rpn operation types.

Like all RPN calculators, rpn uses stacks. In particular, it has a numeric stack, a logical stack, and a string stack. Items are pushed onto the numeric stack whenever a number-token is entered, or whenever an operation concludes that has a number as its result; items are popped from this stack by operations that require numeric arguments. Items are pushed onto the logical stack whenever a logical expression is evaluated; they are popped from this stack by use of logical operations that require logical arguments (e.g., logical ANDing), or by conditional branch instructions. Items enclosed in double quotes are pushed onto the string stack; items are popped from this stack by use of operations that require string arguments (e.g., formatted printing).

rpn supports user-defined memories and functions. To create a user-defined memory, one simply stores a value into the name, as in ``1 sto unity''; the memory is created automatically when rpn detects that it does not already exist. To create a user-defined function, enter the ``udf'' command; rpn will prompt you for the function name and the text that forms the function body. To invoke a UDF, simply type the name.

A file containing rpn commands can be executed by pushing the filename onto the string stack and invoking the ``@'' operator. rpn supports more general file I/O through the use of functions that mimic the standard C I/O routines. Files are identified by integer unit numbers, with units 0 and 1 being permanently assigned to the terminal input and terminal output, respectively.

About this document ...

User's Manual for `elegant`

This document was generated using the **LaTeX**2HTML translator Version 99.2beta6 (1.42)

Copyright © 1993, 1994, 1995, 1996, Nikos Drakos, Computer Based Learning Unit, University of Leeds.
Copyright © 1997, 1998, 1999, Ross Moore, Mathematics Department, Macquarie University, Sydney.

The command line arguments were:
latex2html `elegant`

The translation was initiated by Michael Borland on 2000-04-10

Michael Borland 2000-04-10

Nikos Drakos

Contact Information

Nikos Drakos is no longer employed at the Computer Based Learning Unit of the University of Leeds but can be reached at nikos@mpn.com. The LaTeX2HTML archives will continue to be maintained and will be kept at the usual address at http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html

Interests

Many.

Work Groups

- Graphical Programming and Intelligent User Interfaces

Publications

- Nikos Drakos. **LaTeX2HTML Usage Statistics**. Internal Report, Computer Based Learning Unit, University of Leeds, June 1994.
  (html)
- Nikos Drakos. **From Text to Hypertext: A Post-Hoc Rationalisation of LaTeX2HTML**. To appear in: The Proceedings of the First WorldWide Web Conference, May 1994, CERN, Geneva, Switzerland.
  (html+postscript)
- Nikos Drakos. **Text to Hypertext conversion with LaTeX2HTML**. Baskerville, December 1993, Vol. 3, No. 2, pp 12-15.
  Reprinted with minor modifications in the MAPS TeX Journal.
  (postscript or html)
- **A LaTeX to HTML translator** (postscript or hypertext).
- **Interactive graphical programming environments and software construction** (hypertext).
- **Object Orientation and Visual Programming**. Extended abstract in postscript or hypertext .

More publications...

Nikos Drakos
Wed Nov 17 20:11:11 GMT 1993

You have arrived at...

Ross Moore's Home Page

- Sorry, this is pretty boring at the moment.

(However if you click on the picture you will get the full photo from which it was extracted. ~60k gif)

I am working on other projects and will get around to making this more interesting in due course, RSN.
RSN = ``**r**eal **s**oon **n**ow" ; or

``**R**ather **S**ooner than **N**ever"

``**R**oss is **S**low at **N**ew things"

In the mean time, here are links to some other local web-pages:

- Maths Dept. Picture Gallery
- Vacation Scholars 1995
- Vacation Scholars 1996
- Bush Band
- Mathematical Sculpture
- Jane Austen Society
- Feathered Friends
- Mathematics at Macquarie...Staff
- Mathematics at Macquarie...Current Research

Teaching

Courses 1999, 1st semester

- MATH 130 Mathematics 1E
- MATH 132 Mathematics 1A (Advanced)
- MATH 337 D1 Algebra IIIA (daytime)
- MATH 337 E1 Algebra IIIA (evening)

Research interests

Top of the list, at the moment, is ...

LaTeX2HTML

This program, written in Perl, is for translating documents marked-up using LaTeX-like syntax into well-structured HTML pages.
It works with Unix, Linux, Windows-NT, OS/2 and even DOS.

Originally devised by Nikos Drakos, this has been extended to become a really useful translation tool, producing very nice images of mathematics (for instance), and capable of handling quite intricate technical documents.
See the online manual for details.

The above course descriptions, and the exercise sheets and solutions to which they link, were prepared using this tool.

Here are some more examples, created locally:

- National Symposium in Mathematics
- a Geo-Mathematics paper
- Quantum Groups, see below
- Xy-pic User's Guide, see below

Xy-pic

This is a suite of macros for typesetting mathematical and other technical diagrams, using TeX or LaTeX.

Written originally by Kristoffer Rose, (of DIKU, Denmark) it has undergone extensive revision and extension over the past couple of years.

Basically, Kris provides the computing structures while I contribute mathematical ideas; though sometimes it is the other way round.

Check out the Xy-pic Home Page

Peruse the Xy-pic User's Guide to see the power and versatility of these macros.

From the younger continents in the world, it may be easier to visit Kris Rose's home: Xy-pic at BRICS.

Try here to get the latest version of Xy-pic, version 3.7: local ftp site.

Alternatively a CTAN site may be more convenient:

- by ftp, from a mirror site in Australia: UNSW, Sydney.
- by ftp: Univ of Queensland.
- use the search engine (very convenient!):

.
- or find the one nearest you.

Quantum Groups

This is a book written by a colleague, Ross Street which I have typeset using LaTeX and **LaTeX2HTML**. It is full of categorical diagrams, specified using Xy-pic.

Mathematica Graphics

Some attractive graphics that I have produced using *Mathematica* can be found in: Garnet Surfaces. These reconstruct foliation surfaces, i.e. deformed layers of included minerals within rock-samples, in the presence of crystals of garnet.

Indeed, here is a **QuickTime** movie of such a reconstructed inclusion surface, viewed from different angles, so apparently rotating: Spiral Movie (354k) .

Other uses of *Mathematica* for Geo-physical modelling are described in a Geo-Mathematics paper, to be published in a special volume of *Computers & Geosciences*, late in 1999.

Here is a *Mathematica* Notebook containing frames for a movie of a reconstruction of foliation surfaces forming a ``millipede'' microstructure: Millipede (431k)

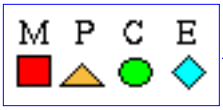
Here are some **QuickTime** movies of the same thing, sectioned in different ways:

- Millipede, YZ-sections (1MB+)
- Millipede, XZ-sections (800k)
- Millipede, XY-sections (820k)

For viewing QuickTime movies on various platforms, I recommend:

- Macintosh: **Simple Player**, or many other utils.
- Unix: XAnim Rev 2.69.7.7,(2 May 95), by Mark Podlipec.
- Unix: XAnim Rev 2.68.5,(23 Aug 94), by Mark Podlipec.
- Windows: **QuickTime for Windows**

Dr Ross R Moore X500
Mathematics Department
Macquarie University Australia 2109
Fax: +61 2 850 8914
Work Phone: +61 2 850 8955
Internet: ross@mpce.mq.edu.au



MPCE Home Page *ross@mpce.mq.edu.au : 18 Jan 1995*

Ross Moore / Mathematics Department / ross@mpce.mq.edu.au